DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

PROYECTO FIN DE CARRERA

DISEÑO DE UNA INTERFAZ GRÁFICA CON MATLAB PARA LA CREACIÓN Y TRANSMISIÓN DE SEÑALES A UN GENERADOR DE FUNCIONES VÍA RED DE ÁREA LOCAL

Autor: Paolo Horna Dueñas

Tutor: Dr. Víctor Pedro Gil Jiménez

Leganés, Octubre 2011

Título: Diseño de una interfaz gráfica con Matlab para la creación y transmisión de señales a un generador de funciones vía Red de Área Local.

Autor: Paolo Horna Dueñas

Tutor: Dr. Víctor Pedro Gil Jiménez

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día de Octubre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar empezar dando las gracias a Dios, que ha permitido que tenga la oportunidad de estudiar esta carrera, y quién a su vez me ha dado fuerzas y aliento para seguirla. Él me ha dado paciencia, perseverancia, capacidad de sacrificio, y sobre todo amor y pasión por lo que hago, cosas fundamentales a la hora de emprender este camino, y cualquier otro, ya que sin ellos todo lo que se hace carece de fundamento.

En segundo lugar agradecer también a mi familia quienes me han apoyado en todo momento, dándome ánimo en momentos de flaqueza, y sobre todo teniendo comprensión por lo que hacía. Y especialmente a mi hermanita Victoria de tres años, quién me ha hecho mucho más agradable estos últimos años de estudio.

Por supuesto agradecer también a mi tutor Víctor, que me ha dedicado muchos momentos de atención y resuelto muchas dudas a lo largo del desarrollo de este proyecto, y no sólo agradecer a él sino también a todos los profesores que me han enseñado mucho.

Glosario

LAN	Local Area Network (Red de Área Local)
GPIB	General Purpose Interface (Interfaz de Propósito General)
USB	Universal Serial Bus (Bus Universal en Serie)
FFT	Fast Fourier Transform (Transformada Rápida de Fourier)
AM	Amplitude Modulation (Modulación de Amplitud)
FM	Frecuency Modulation (Modulación de Frecuencia)
PM	Phase Modulation (Modulación de Fase)
FSK	Frecuency Shift Keying (Modulación por desplazamiento de frecuencia)
PWM	Pulse Width Modulation (Modulación por ancho de pulsos)
VISA	Virtual Instrument Software Architecture (Arquitectura de Instrumentos de Software Virtual).

Resumen

El objetivo de este proyecto es crear un programa en Matlab que se encargue de enviar señales vía Red de Área Local (LAN) a un generador de funciones Tektronix AFG3102.

Los generadores de funciones arbitrarios Tektronix AFG3102 permiten generar cualquier señal de frecuencia inferior a 100 MHz que previamente haya sido diseñada por el usuario, se utilizan en diversos laboratorios de las titulaciones de telecomunicaciones, pero están muy limitados por el software que nos sirve como interfaz, el ArbExpress®.

Por ello se va a realizar un programa que se encarga de enviar las señales deseadas por el usuario, además este programa contará con una interfaz gráfica en la cual el usuario además de seleccionar el tipo de onda podrá elegir todos los parámetros que caracterizan a las señales como son la frecuencia, la amplitud, la fase, el offset, etc.

Abstract

The objective of this project is to create a program in Matlab which is responsible for sending signals via Local Area Network (LAN) to a Tektronix AFG3102 function generator.
Generators Tektronix AFG3102 arbitrary function can generate any frequency signal below 100 MHz that has been previously designed by the user, are used in various laboratories of the telecommunications degree, but they are limited by the software that serves as an interface, the ArbExpress®.

For it is here to make a program that is responsible for sending the signals desired by the user, and this program will have a graphical interface in which users also select the type of wave can choose all the parameters that characterize the signals such as frequency, amplitude, phase, offset, etc.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción y Objetivos

1.1 Introducción

El hombre, desde sus orígenes se caracteriza por fabricar toda clase de herramientas que le ayudan en su vida cotidiana. Básicamente cuando se fabrica una herramienta se tiene en mente qué es lo que va a desempeñar, qué es lo que se va a medir exactamente. Una vez fabricado, el objeto sirve para lo que se ha diseñado, tiene una función y no sirve para otra cosa.

Un instrumento virtual es un utensilio de medida cuya función no está definida permanentemente. Unas veces sirve para medir una determinada propiedad o de una determinada manera y otras veces, según se programe se podrá utilizar para medir otra cosa, o de otra determinada manera.

Por ejemplo, y centrándonos en el ámbito electrónico, un osciloscopio es un instrumento de medida en el sentido clásico. Con él podemos ver tensiones, formas de onda, e incluso FFTs (Transformadas de Fourier). Bien, pero no podemos medir un diagrama de BODE directamente.

Sin embargo, si asociamos un osciloscopio, un generador de señales y un mecanismo de control de ambos, como un ordenador programable y una serie de interfaces, podemos realizar una medida en la que colaboren todos los instrumentos para obtener por pantalla, en el ordenador, un diagrama de BODE, por ejemplo.

Y luego cambiar de programa y medir otra cosa, como el desfase entre dos señales, la distorsión armónica, etcétera, sin cambiar ni un solo componente *hardware*.

A este conjunto de instrumentos *clásicos*, con funciones y finalidades cambiantes es a lo que se denomina *instrumento virtual.*

Descripción del instrumento: El instrumento que se ha desarrollado como primer puesto de instrumentación virtual, consiste en:

- 1 osciloscopio Tektronix TDS5034B
- 1 generador de señales Tektronix AFG3102
- 1 ordenador personal
- Conexión LAN (Red de Área Local)
- 1 cable de conexión entre osciloscopio y el generador de señales.

1.2 Objetivos

El principal objetivo de este proyecto es poder transmitir correctamente desde el PC al generador de funciones diversos tipos de señales a través de la conexión LAN usando para ello la programación en Matlab, como complemento de este programa se creara una interfaz gráfica con la cual el usuario interactuará con el programa, permitiendo así mas fácil el uso de este programa por cualquier tipo de persona sin necesidad de tener conocimientos de programación.

1.3 Fases del Desarrollo

El desarrollo de este proyecto se divide en diferentes fases:

Fase 1: Descripción del generador de funciones

Fase 2: Descripción del osciloscopio digital usado

Fase 3: Estudio de los diferentes tipos de conexiones con el generador de funciones.

Fase 4: Introducción a Matlab y descripción de cómo se crea una GUI (Interfaz Gráfica) con Matlab.

Fase 5: Implementación del programa y de la interfaz grafica

Fase 6: Interacción del programa y el generador de funciones

Fase 7: Envío y transmisión de las señales al generador.

Fase 8: Prueba del funcionamiento del programa

Fase 9: Extracción de conclusiones

1.4 Medios usados

La herramienta que se ha utilizado para realizar este proyecto ha sido Matlab®.

1.5 Estructura de la memoria

Para poder facilitar la lectura y entendimiento de este proyecto a continuación se describe un breve resumen de cada capítulo del proyecto.

Capítulo 1

En este capítulo se presenta una breve introducción del proyecto, así como los objetivos a seguir durante el mismo, y también se explican las fases del desarrollo y los medios usados para la creación del mismo.

Capítulo 2

Este capítulo se encarga de describir uno de los equipos usados como es el generador de funciones, que será el encargado de recibir las señales enviadas por el programa creado.

Capítulo 3

Este capítulo realiza una pequeña descripción del osciloscopio usado y de los usos que se pueden hacer conectando el generador de funciones con éste.

Capítulo 4

Aquí se describirá el interfaz usado para la conexión existente entre el PC y el generador de funciones, llamado TekVISA®, éste se puede usar mediante USB (Universal Serial Bus -Bus Universal en Serie), GPIB (General Purpose Interface - Interfaz de Propósito General) y LAN (Local Area Network - Red de Área Local), este último será el usado para este proyecto.

Capítulo 5

Aquí se entrará a contar ampliamente el medio usado para la creación del programa de envío de señales, que es Matlab, también se describirá como se crea una interfaz gráfica usando este lenguaje de programación.

Capítulo 6

En este capítulo se centra toda la creación del proyecto, ya que este capítulo entra en detalle en la forma de cómo se ha creado el programa con interfaz gráfica y los pasos a realizar para su funcionamiento.

Capítulo 7

Aquí se recogen las conclusiones a las que se han llegado después de la realización del proyecto.

Capítulo 8

Muestra el coste de este proyecto.

Capítulo 9

Aquí se reflejan todas las referencias empleadas.

Capítulo 2

El Generador de Funciones Tektronix AFG3102

2.1 Características de los generadores Tektronix

A continuación describiré las principales características de los diferentes generadores de funciones Tektronix con la siguiente tabla [1].

Modelo	**AFG3102**
Canal	1/2
Sinusoidal	100 MHz
Pulso	50 MHz
Memoria	>16384 a 131072
Velocidad de muestreo	250 MS/s
Amplitud	10 V_{p-p}
Pantalla	Color
Interfaz	USB, LAN, GPIB

Tabla 2.1. Características generales del Generadores de funciones Tektronix modelo AFG3102.

Estos generadores tienen tres funciones integradas en un generador:

- 1 Generador de funciones hasta 100 MHz
- 1 Generador de pulsos hasta 50 MHz
- 1 Generador de formas de onda arbitrarias de 14 bits.

Más características del generador:

- Pantalla LCD a color monocromática
- Funcionamientos síncrono
- Interfaz USB

Descripcion Física del Generador

Seguidamente se realizará una descripción física del generador mediante la muestra de dos imágenes correspondientes a las dimensiones y a la configuración de botones del aparato.

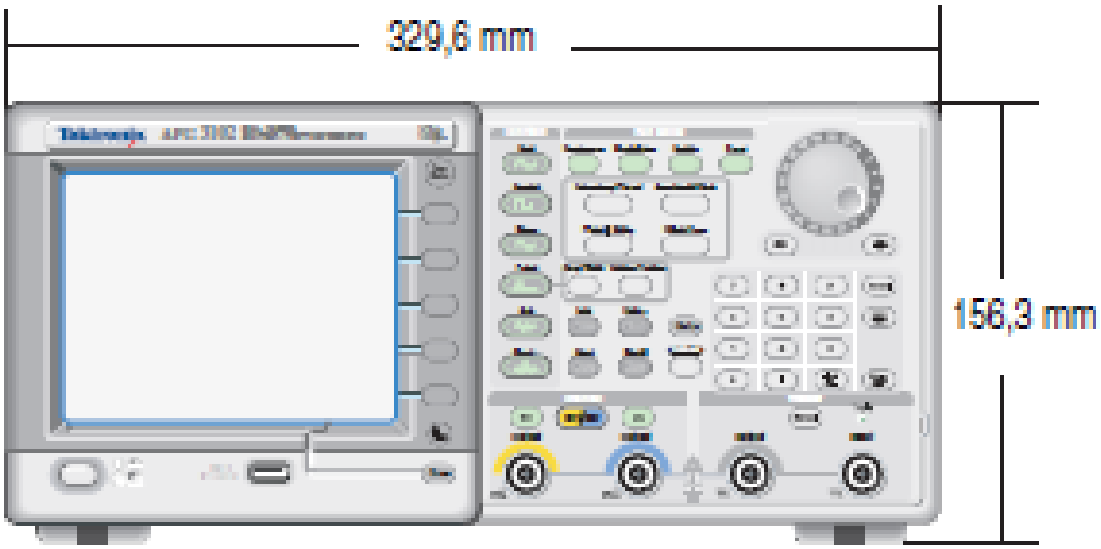


Figura 2.1 .Dimensiones del generador Arbitrario,[1].

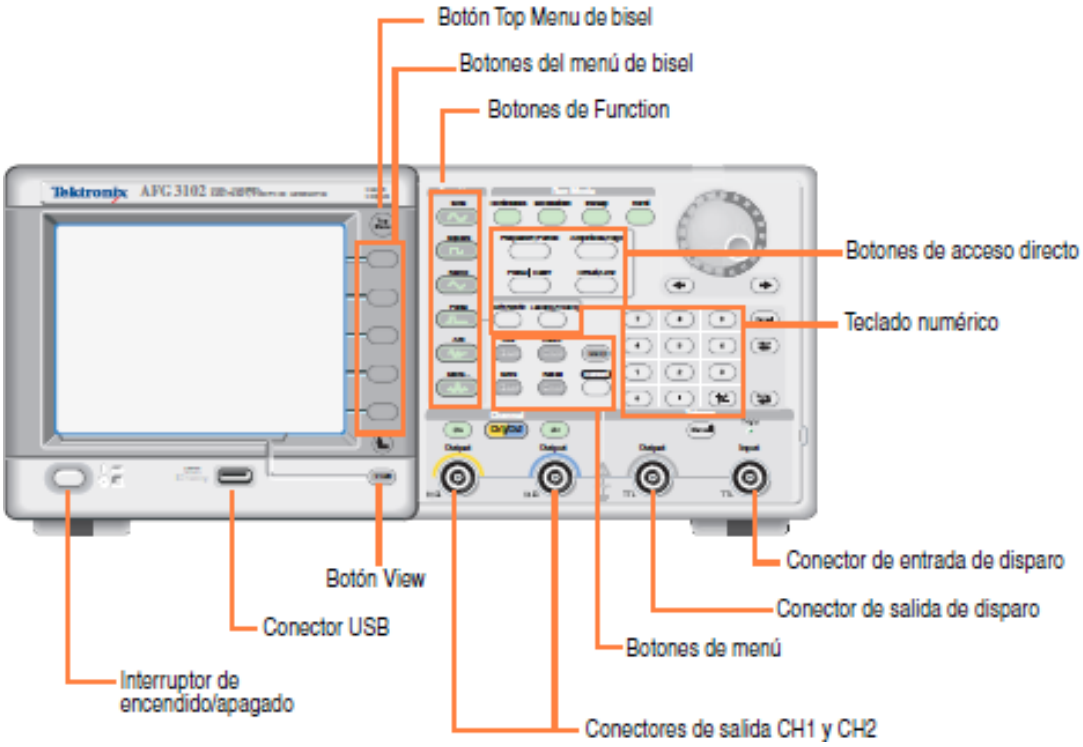


Figura 2.2. Configuración de los botones,[1].

2.2 Selección de una forma de Onda

El instrumento puede proporcionar 12 formas de onda estándar (Seno, Cuadrada, Triangular, Pulso, Sinc, Ruido, DC, Gaussiana, Lorentz, Exponencial creciente y decreciente y Haversine).

Este también puede proporcionar formas de onda definidas por el usuario, puede crear y editar formas de onda personales.

También se puede crear formas de onda moduladas utilizando los menús de la opción *Modulation* de *Run Mode*.

La tabla 2.2 muestra el tipo de combinación de modulación soportada para cada forma de onda salida[1].

	Seno, Cuadrada, Triangular, Sinc, Gaussiana, Lorentz, Exponencial creciente, Exponencial decreciente y Haversine	**Pulso**	**Ruido y DC**
AM	Sí		
FM	Sí		
PM	Sí		
FSK	Sí		
PWM		Sí	
Barrido (Sweep)	Sí		
Ráfaga (Burst)	Sí	Sí	

Tabla 2.2 Combinación de los tipos de modulación y la forma de onda salida

2.3 Introducción al uso del generador

PASOS PARA SELECCIONAR UNA FORMA DE ONDA DE SALIDA:

1) Para seleccionar una forma de onda sinusoidal continua, se pulsará el botón *Sine* del panel frontal, y a continuación, el botón *Continuous***.**

2) Se puede seleccionar directamente una de las cuatro formas de onda estándar con los botones de función (*Function*) del panel frontal.

3) Para seleccionar una forma de onda arbitraria, se pulsará el botón *Arb*.

4) Para seleccionar otras formas de onda como Sinc (Sin(x)/x), Ruido (Noise), DC o Gaussiana (Gaussian), se pulsará el botón *More***...** y luego el botón del bisel superior correspondiente.

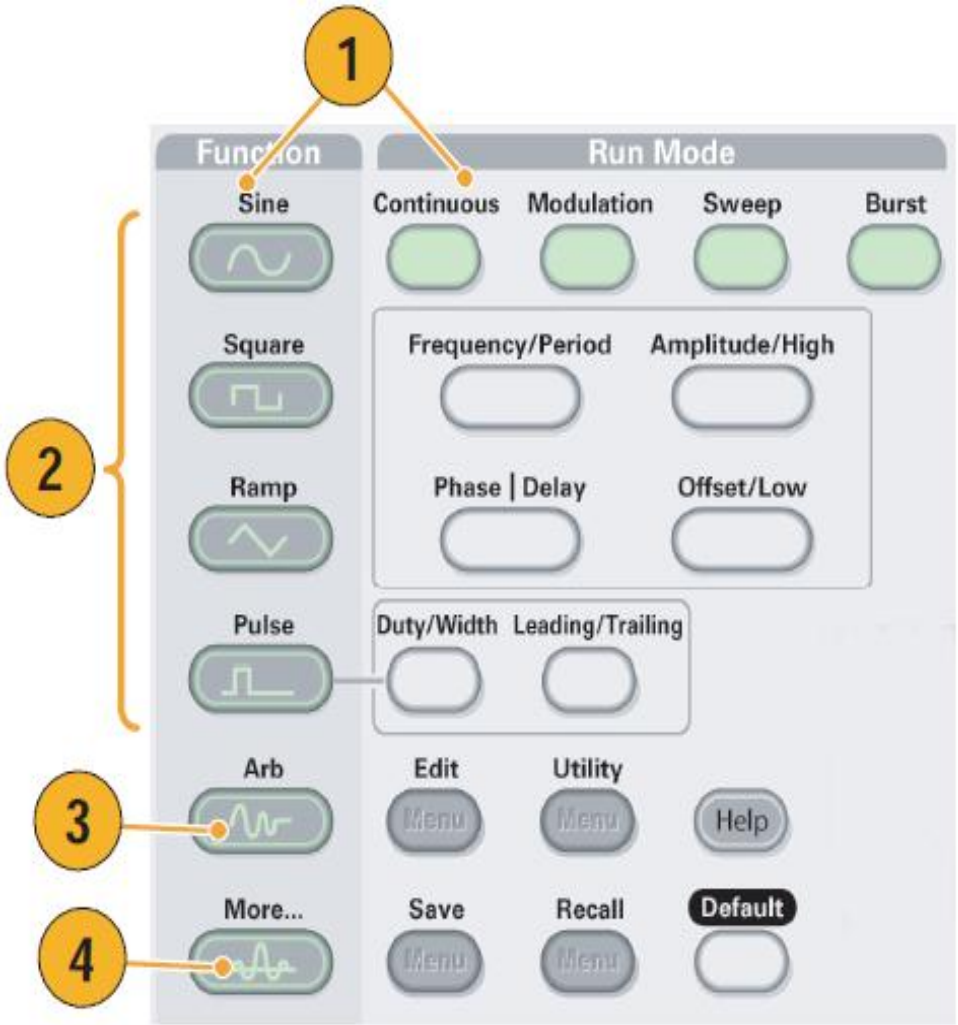


Figura 2.3. Panel principal del generador de funciones, [1].

EJEMPLOS DE FORMAS DE ONDAS:

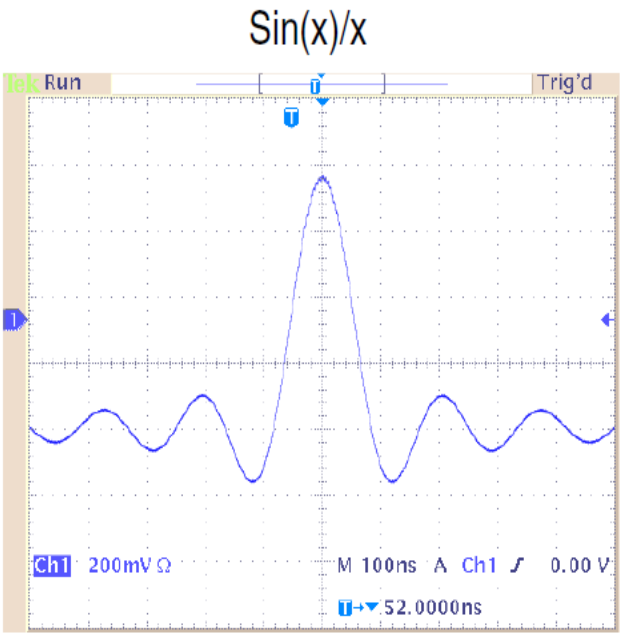


Figura 2.4. Gráfica de una señal Sinc, [1].

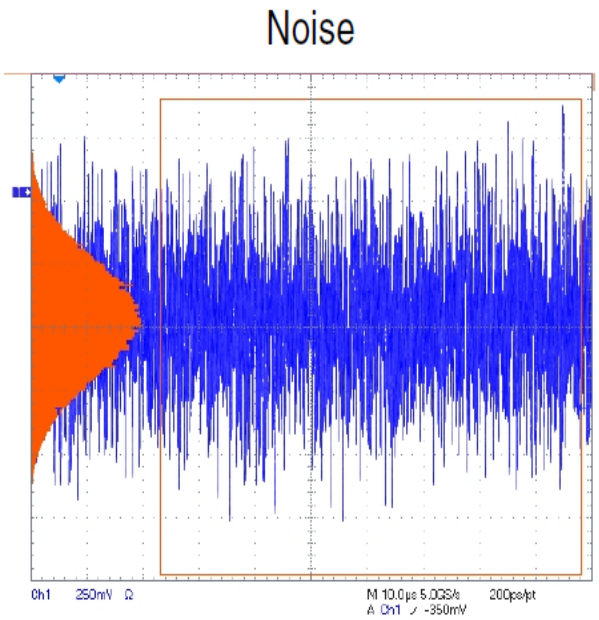


Figura 2.5. Gráfica de una señal Ruido, [1].

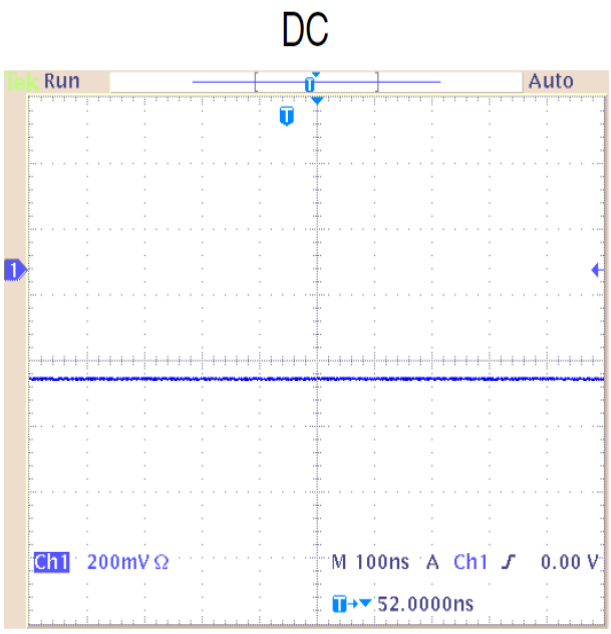


Figura 2.6. Gráfica de una señal DC, [1].

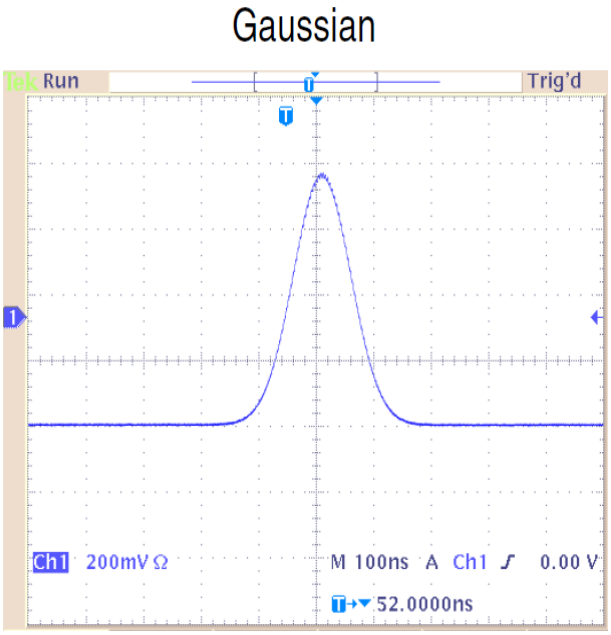


Figura 2.7. Gráfica de una señal Gaussiana, [1].

MÁS EJEMPLOS DE FORMAS DE ONDAS:

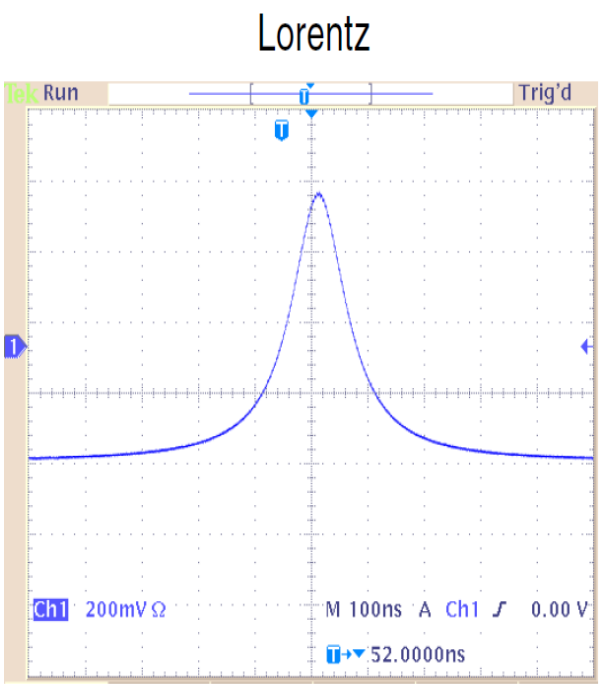


Figura 2.8. Gráfica de una señal Lorentz, [1].

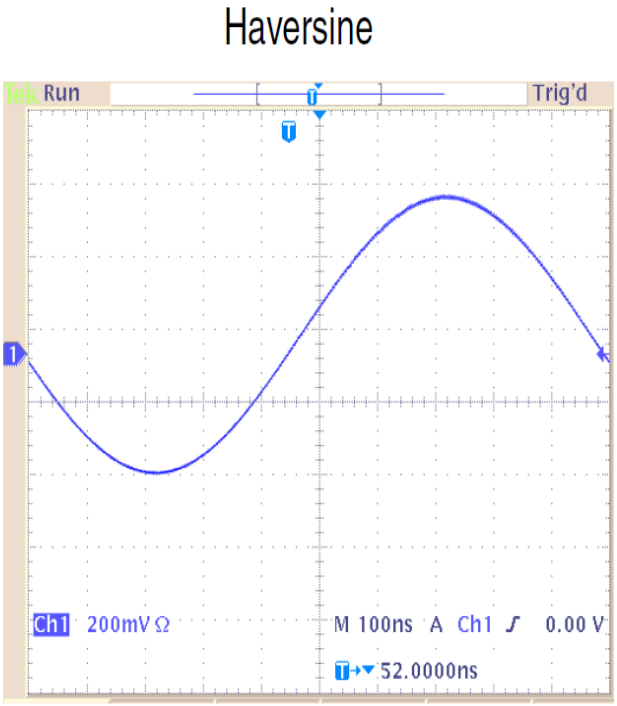


Figura 2.9. Gráfica de una señal Haversine, [1].

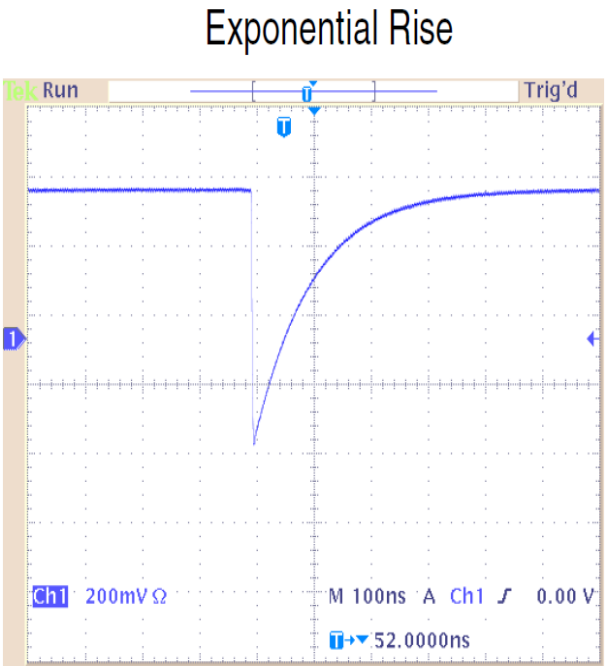


Figura 2.10. Gráfica de una señal Exponen. Creciente, [1]

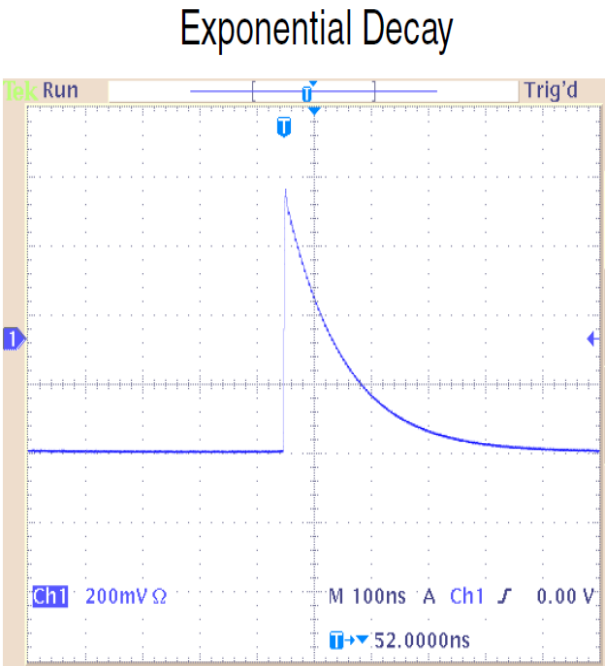


Figura 2.11. Gráfica de una señal Exponencial Decreciente, [1].

Capítulo 3

El Osciloscopio

Tektronix TDS5034B

3.1 Características de los Osciloscopio Tektronix

Después de enviar la señal al generador de señales conectaremos el generador con el osciloscopio, para apreciar mejor las señales y poder modificarlas más ampliamente.

Las que siguen son las principales características del osciloscopio.

PARA ANALIZAR LAS FORMAS DE ONDA DISPONEMOS DE LAS SIGUIENTES TOMAS DE MEDIDA Y ESTOS SON LOS PASOS A SEGUIR:

1) Selección de las medidas.

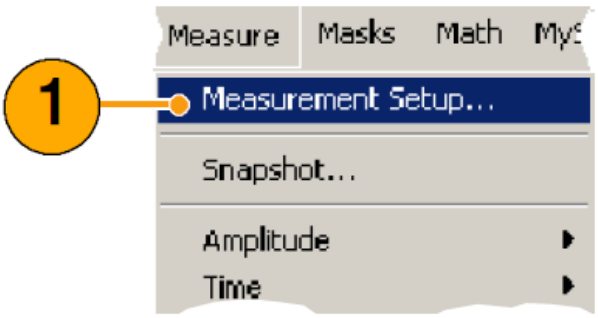


Figura 3.1 Menú de selección de medidas del osciloscopio, [2].

2) Selección del canal, math o referencia de la forma de onda que se desea medir.

3) Usando las pestañas de selección tenemos ocho medidas dentro de las cinco diferentes categorías.

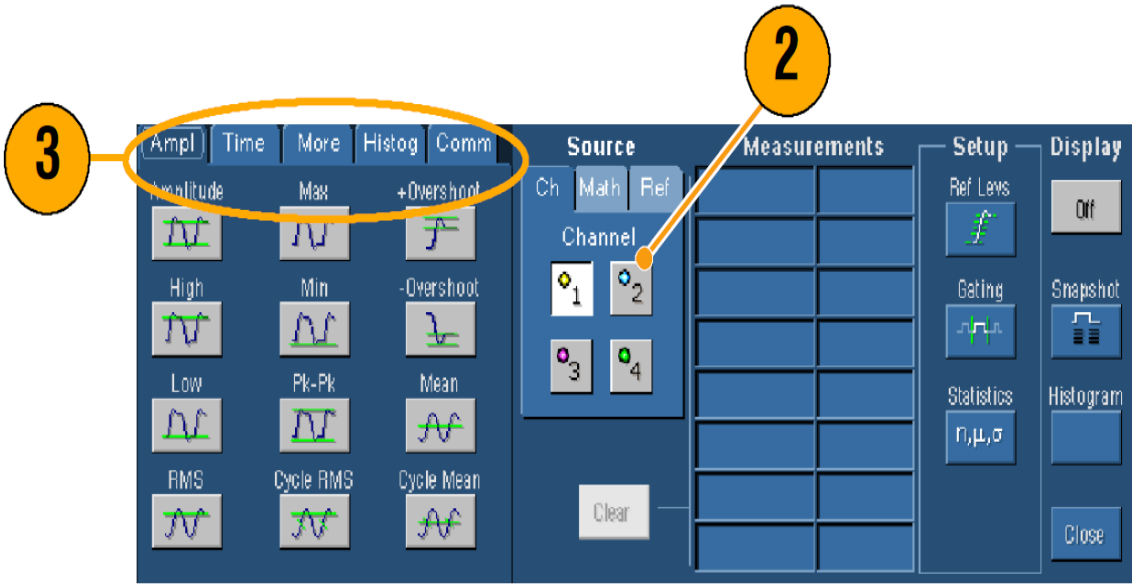


Figura 3.2 Pantalla principal del osciloscopio, [2].

4) Para borrar la última medida pulsamos CLEAR.

5) Para deshacer las medidas últimas realizadas pulsamos en CLEAR ALL.

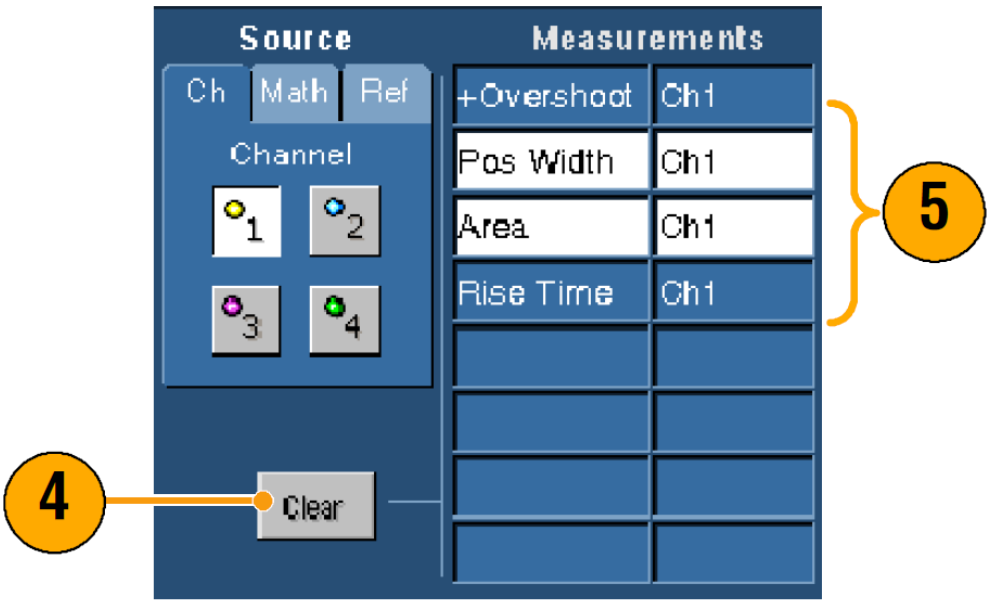


Figura 3.3 Pantalla de selección de canal del osciloscopio, [2].

Si se desea también se puede escoger una media para la forma de onda seleccionadas directamente en el menú Medida (*Measure*).

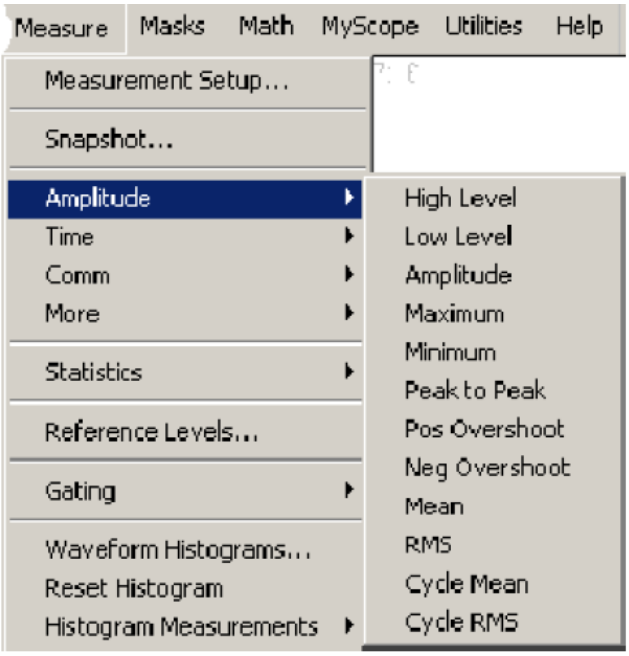


Figura 3.4 Menú Measure de selección de medidas del osciloscopio, [2].

INFORMACIÓN AUXILIAR BREVE:

En el *ROLL MODE* las medidas no están disponibles hasta después que se para la adquisición de la señal.

Para añadir más medidas, hacemos clic en la forma de onda y clic después en añadir medida.

Para borrar una medida, *REMOVE.*

Para borrar todas las medidas realizadas, *REMOVE ALL.*

POSIBLES USOS DE LAS SEÑALES RECIBIDAS POR EL OSCILOSCOPIO

Algo muy usado es el **análisis espectral de las señales**, y los pasos los definimos a continuación.

1) Se selecciona el math.

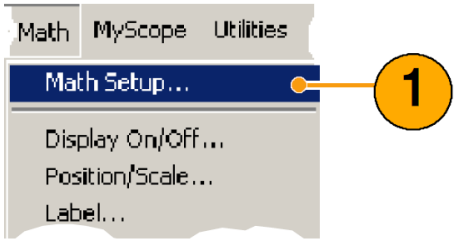


Figura 3.5 Menú Math de selección del osciloscopio, [2].

2) Se elige una de las predefinidas expresiones math espectrales.

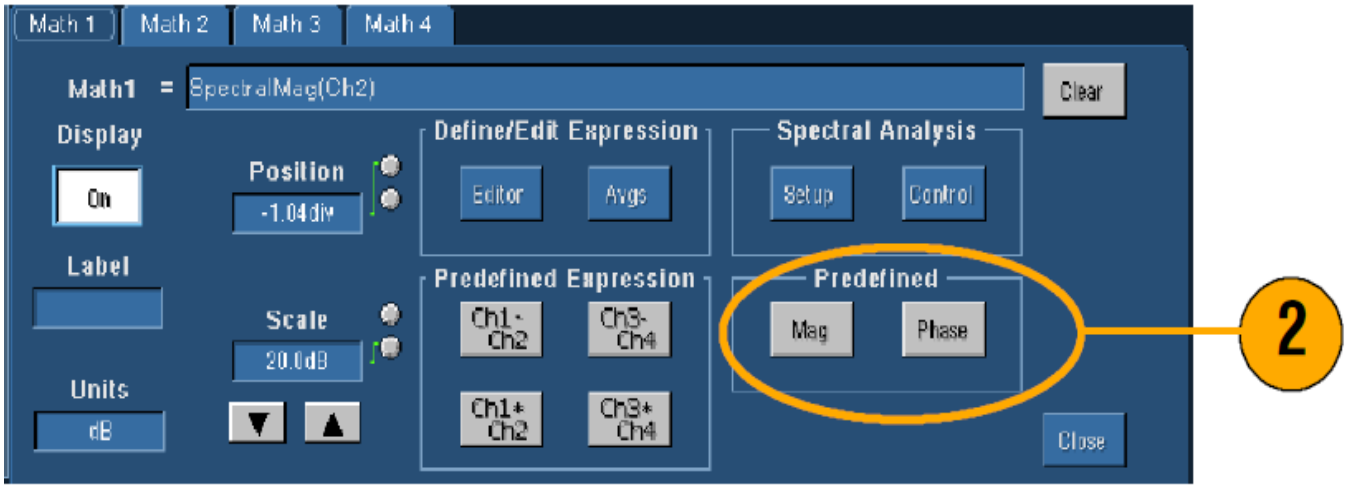


Figura 3.6 Menú para la selección de expresiones math espectrales, [2].

Este es el **procedimiento para construir un espectro:**

1) Selección de Math - Espectral setup.

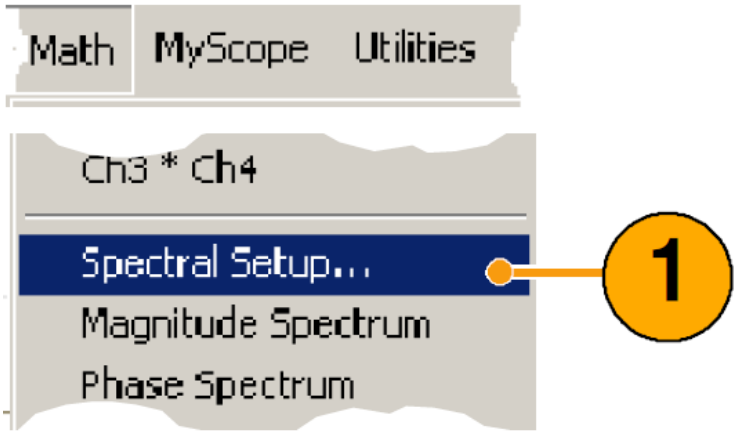


Figura 3.7. Opción Spectral Setup, [2].

2) Selección de la forma de onda.

3) Selección del tipo de onda espectral que se desea crear.

4) Selección del origen de la onda.

5) Ajustar la forma de onda espectral.

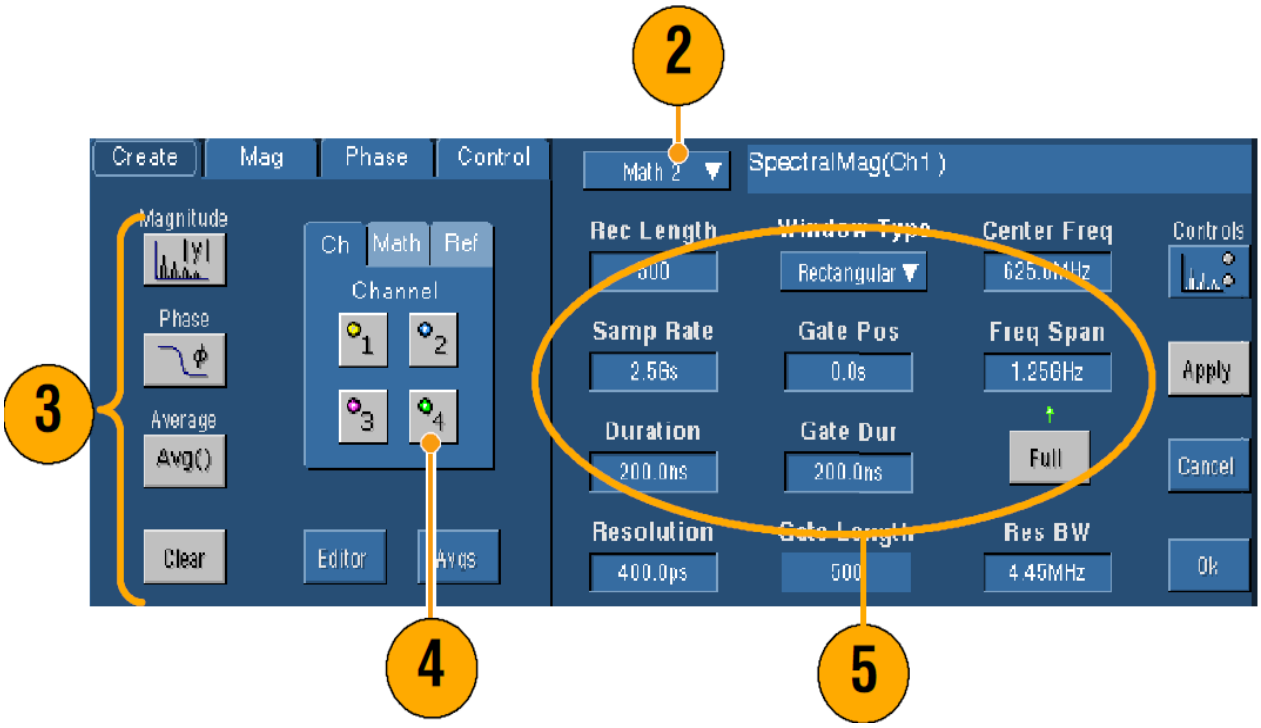


Figura 3.8 Panel de selección de forma de Onda, tipo de onda espectral y canal de origen, [2].

El ajuste de la forma de onda espectral se realiza con:

1º Spectral Setup Control Window.

2º Después se pulsa en CONTROLS y ajustar la forma de la onda.

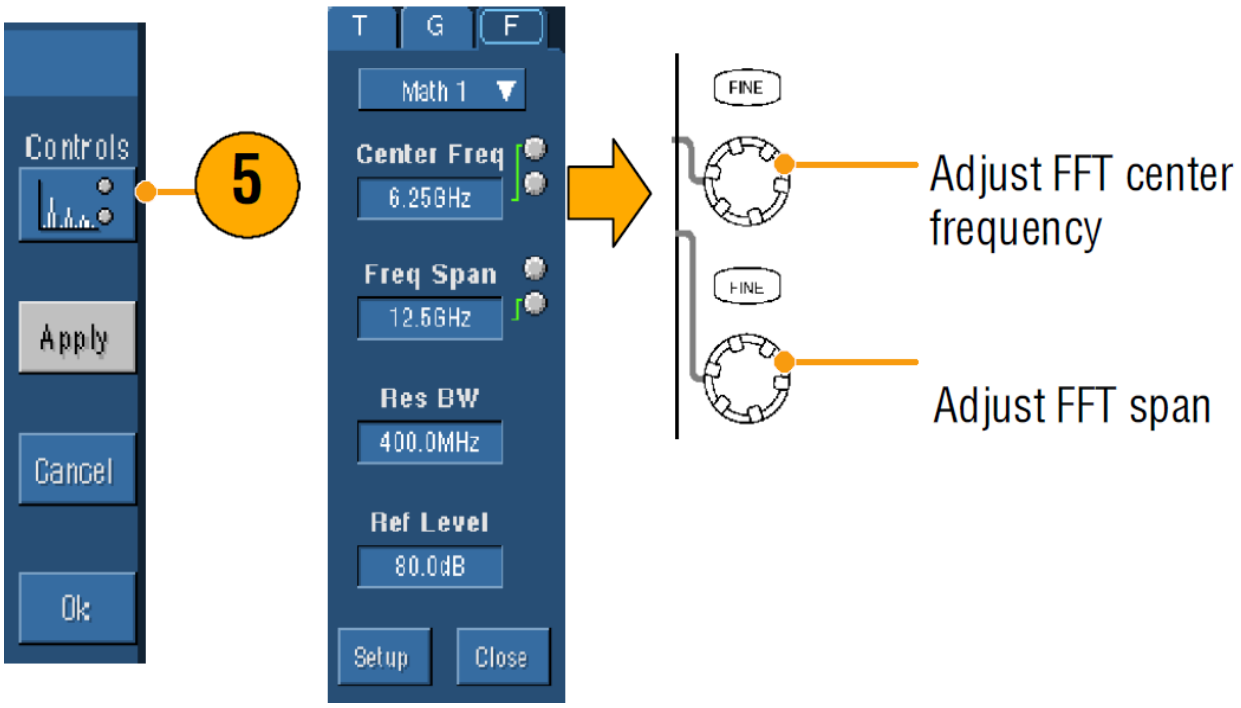


Figura 3.9 Ajuste de la forma de Onda espectral, [2].

6) Si se desea también se puede ver el dominio en tiempo y frecuencia de las formas de onda a la vez.

Para ello se usa *GATING* que sirve para seleccionar la porción de le onda en el dominio del tiempo para el posterior análisis espectral.

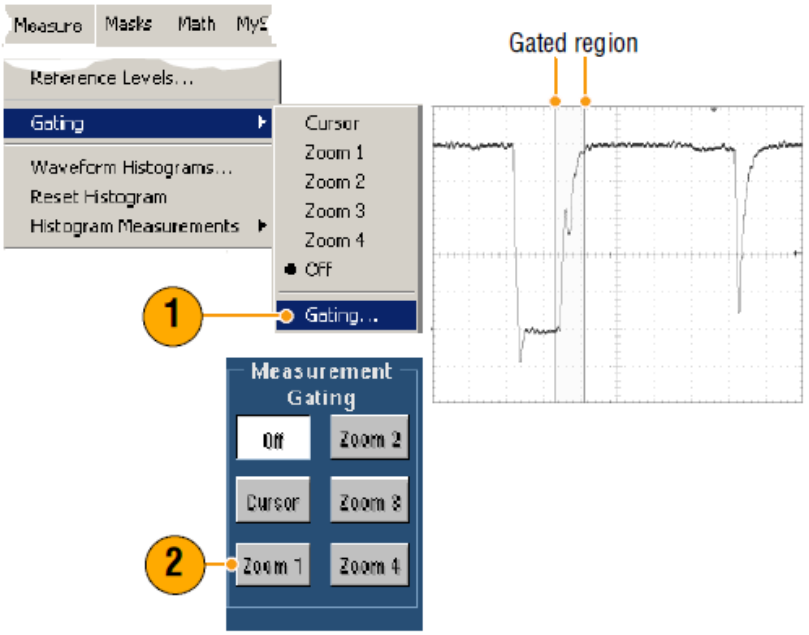


Figura 3.10 Uso del *Gating*, [2].

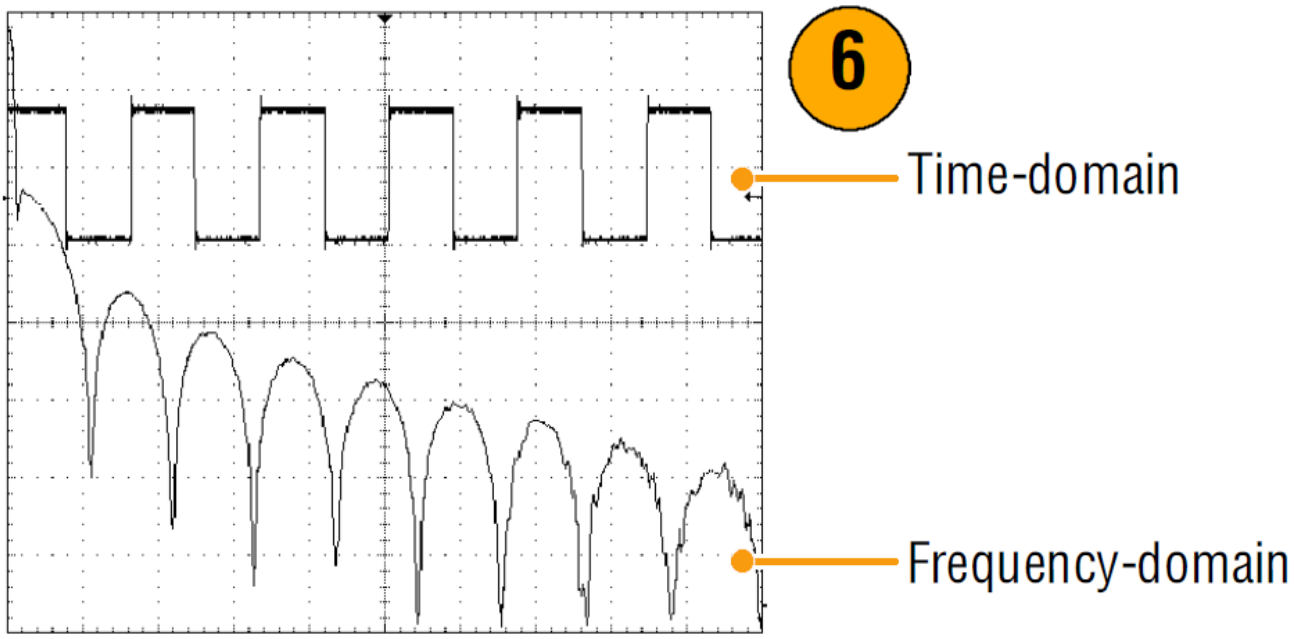


Figura 3.11 Gráfica de la Onda en el dominio del tiempo y frecuencia, [2].

Capítulo 4

Interfaz TekVISA® para la conexión entre el PC y el Generador

4.1 Descripción de la interfaz TEKVISA

La conexión entre el ordenador personal y el generador de funciones se realiza a través de una interfaz de terminal llamada TekVISA® que es proporcionada por la marca Tektronix®.

Las aplicaciones de prueba y medición requerían de algún tipo de librería de I/ O (Input/output) para poder comunicarse con instrumentos de prueba.

Como paso previo a toda la industria del software con compatibilidad, la VXIplug & play Alianza de Sistemas desarrolló una librería de Entrada/Salida comunes llamada VISA (Virtual Instrument Software Architecture), que quiere decir Arquitectura de Instrumentos de Software Virtual.

VISA proporciona un estándar común para los desarrolladores de software para que el software de múltiples proveedores, tales como controladores de instrumentos, se puedan ejecutar en la misma plataforma.

Un controlador (driver) de instrumentos es una librería de funciones que se encarga de los detalles de controlar y comunicarse con un instrumento específico, como un Osciloscopio Tektronix.

TekVISA es la aplicación de Tektronix de la VISA Application Programming Interface (API). TekVISA es compatible con la industria software, disponible con la selección de modelos de instrumentos de Tektronix

TekVISA implementa un subconjunto de la versión 3.0 de la especificación de VISA para el control GPIB, USB y serie (RS-232) interfaces de instrumentos a nivel local o de forma remota mediante una conexión Ethernet LAN.

TekVISA proporciona la interfaz independiente con funcionalidad necesaria para el control y acceso al software integrado de Tektronix® con equipos de prueba y medición de las siguientes maneras:

- El uso de software virtual GPIB se ejecutan localmente en sobre instrumentación basada en Windows®.
- El uso físico de un controlador de hardware para GPIB.

- El uso de un controlador de hardware para serie asíncrona.

- Sobre una red de área local (LAN) que utiliza el protocolo VXI-11, TCP / IP Socket, y uno de los siguientes:
    - Un AD007 LAN-GPIB adaptador a GPIB controlador de hardware
    - Una conexión Ethernet con el servidor VXI-11 basado en Windows, tales como TDS7000 y TDS/CSA8000 series de osciloscopios.
    - Una conexión Ethernet con servidor de Socket que se ejecutan en Windows para la instrumentación basada en la serie de osciloscopios como los modelos DPO/DSA7000 de Tektronix.
    - Mediante una conexión USB para los instrumentos con USB, como el TDS1000B y TDS2000B.
    - El uso de hardware TekLink a TekLink habilitados para instrumentos.

4.2 Características y Beneficios de TekVISA

Mejora la facilidad de uso para los usuarios finales mediante una metodología coherente para el uso de controladores de instrumentos a partir de una variedad de proveedores.

- Proporciona librerías de lenguajes para programadores que utilizan múltiples Aplicación de los entornos de desarrollo como se muestra en la figura de después, que incluye:

    - Microsoft C / C + +.
    - Microsoft Visual Basic.
    - LabVIEW software de gráficos utilizando el lenguaje de G.
    - MATLAB software de análisis.
    - Proporciona una utilidad Instrument Manager para configurar y buscar recursos adicionales VISA.
    - Proporciona servicios de depuración, como TalkerListener y CallMonitor.
    - Permite la instalación de software en cualquier número de ordenadores.

4.3 Aplicaciones y Soporte de conectividad de Tekvisa

TekVISA es beneficioso en una gran variedad de situaciones y aplicaciones:

- Un solo controlador puede ser utilizado por múltiples aplicaciones en diferentes entornos de desarrollo.
- Los controladores de instrumentos de varios fabricantes pueden ser combinados en una única aplicación.
- Los programas de usuario que se ejecutan en instrumentación basada en Windows pueden utilizar TekVISA para controlar el funcionamiento del equipo a través de una conexión virtual de software GPIB, sin utilizar ningún hardware externo GPIB.
- Los programas de usuario que se ejecutan en equipos remotos conectados en red a instrumentación basada en Windows puede utilizar TekVISA para controlar el funcionamiento de instrumentos a través de un GPIB-LAN virtual del servidor VXI-11, y conexión con el servidor Socket no se necesita GPIB-LAN hardware externo. Sólo requiere una conexión Ethernet LAN.
- Los programas de usuario conectado de forma local o remota a otros no basados en Windows con Instrumentación de Tektronix puede utilizar TekVISA para controlar el funcionamiento de instrumentos a través de una GPIB, USB, o en serie (RS232) de conexión local o remota a través de TCPIP directamente o a través de un Tektronix AD007 GPIB-adaptador de LAN.

Las siguientes figuras ilustran la variedad de software, hardware local, y conexiones de red a la instrumentación integrada con el soporte de TekVISA [3].

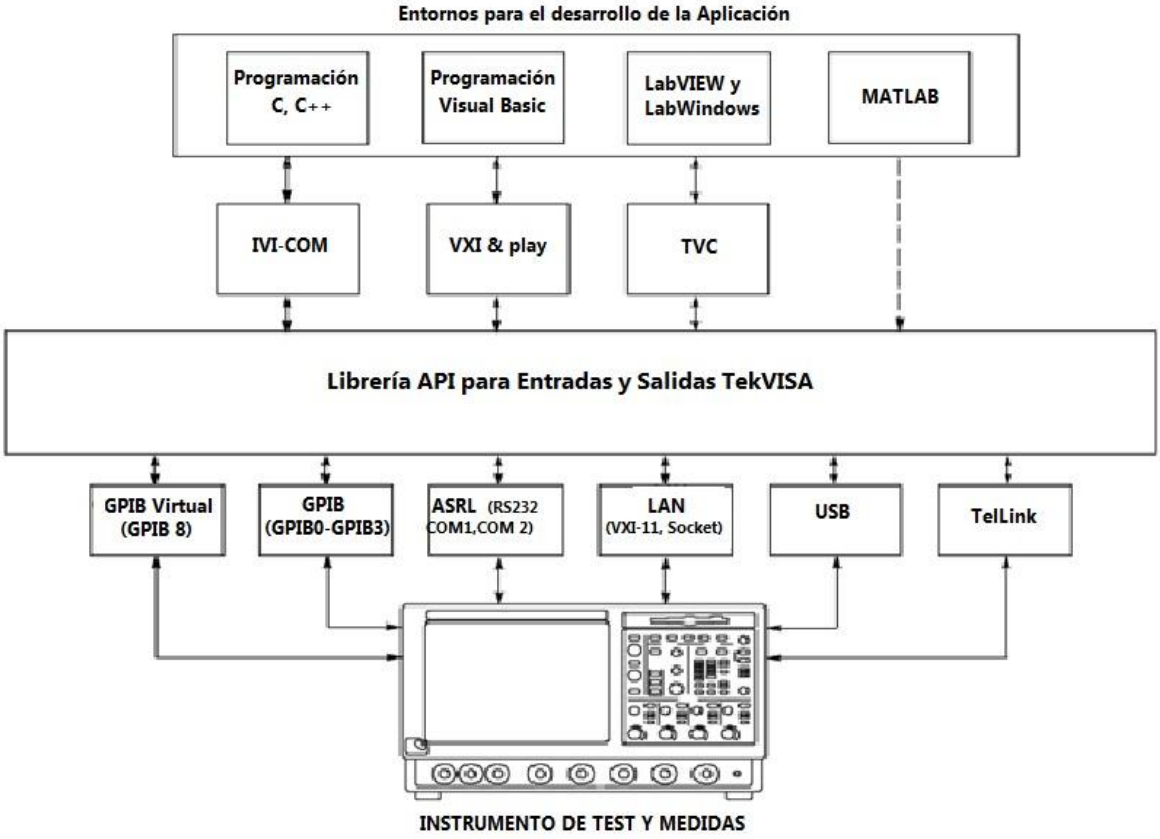


Figura 4.1 Soporte de TekVISA para múltiples entornos de desarrollo, [3].

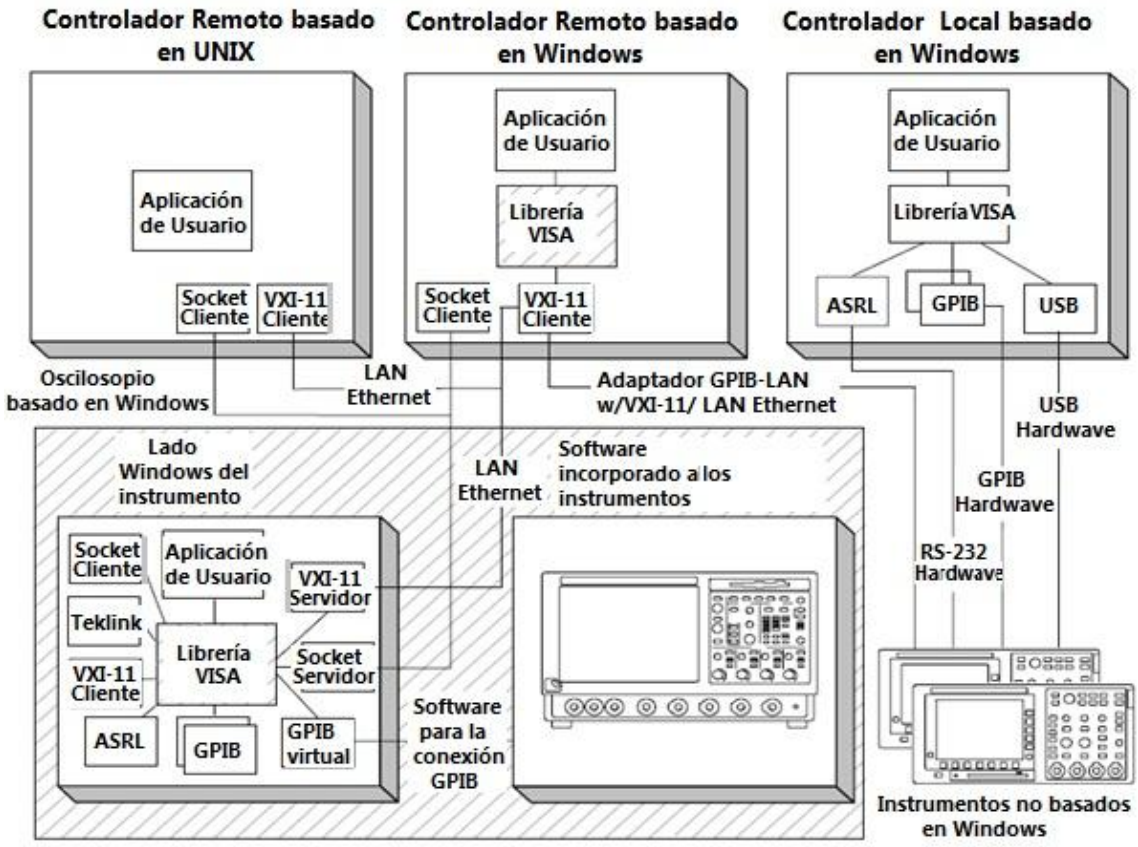


Figura 4.2. TekVISA Soporta conectividad local y remota, [3].

4.4 Conexión del PC con el generador de funciones mediante comandos en Matlab

En este apartado se describe cómo utilizar el software MATLAB para conectarse a los generadores Tektronix [4][5].

Para utilizar el soporte de MATLAB se debe:
1º Asegurarse de que se tiene instalada la versión 6 de MATLAB, versión 12, o posterior.
2º Asegurarse de que se tiene instalado **TekVisa**.

3º Asegurarse de que la ruta del sistema incluye la ruta donde está instalado ArbEther.dll. Por defecto, este archivo DLL se copia en el directorio donde está instalado ArbExpress. Se necesita esta DLL si se conecta al generador a través de Ethernet.

Esta utilidad ArbExpress MATLAB se ha instalado, por defecto, en el camino-ArbExpress \ Tools \ Matlab. El directorio contiene un archivo". P" para cada archivo de comandos y una correspondiente". M" archivo que tiene la documentación para el comando. Un script de ejemplo MATLAB, "Sample.m" lo proporciona Tektronix para mostrar cómo utilizar los comandos en un entorno de esta utilidad.

La secuencia de comandos de MATLAB se almacena en un archivo. M que se crea utilizando una función de MATLAB editor. Al compilar un archivo. M, se obtiene un archivo. P.

Los siguientes archivos deben estar presentes en el directorio antes de empezar a usar los comandos documentados aquí:
Comandos:

- NewSession.p
- CloseSession.p
- query.p
- Read.p
- Write.p
- LoadWfm.p
- TransferWfm.p

Ejemplo de archivo de script:

sample.m

DLL con la rutina de puerta de enlace:

matarb.dll

4.4.1 Transferencia de Archivos desde Matlab

Para transferir archivos de MATLAB, hay que seguir estos pasos.

1º. Iniciar MATLAB.

2º. Ir al directorio donde ha guardado los archivos de librería de MATLAB que se suministran con ArbExpress (C: \ Archivos de programa \ Tektronix \ ArbExpress \ Tools \ Matlab).

Figura 4.3. Ruta de los archivos de MATLAB [4].

3º Abrir una sesión en un generador conectado ya sea a través de LAN, GPIB o USB con el PC:

Para el presente proyecto se usará la conexión a través de LAN, aunque también se describen los demás tipo de conexiones.

Conexión a través de GPIB para la serie de generadores AWG de Tektronix:

```
[SessionID, StatusMsg] = NewSession ('GPIB0:: 9:: INSTR', 'GPIB')
```

Este comando solo trabaja si el "Tekvisa Resource Manager", que es el software que se encarga de reconocer los dispositivos Tektronix conectados al PC, ha detectado los instrumentos.

Conexión a través de LAN con el comando NewSession para la serie de generadores AFG3000 de Tektronix:

```
[SessionID, StatusMsg] = NewSession ('TCPIP : : 10.20.30.40 : : INST',
'LAN')
```

Conexión a AFG300, AWG2000, AWG400/500/600/700 series sin usar Visa (el soporte se realiza a través de conectores directos):

```
[SessionID, StatusMsg] = NewSession ('10 .20.30.40 ',' TCPIP ')
```

Conexión a través de USB:

Para conectarse a un generador AWG Tektronix con el comando NewSession a un instrumento conectado a través de USB se hace de la siguiente manera:

```
[SessionID, StatusMsg] = NewSession ('USB0:: 0x0699:: 0x0343::
JU010107:: INSTR', 'USB')
```

Si el comando se ejecuta correctamente, el identificador de sesión ID está disponible en la variable SessionID y el StatusMsg es una cadena nula. Si el comando no tiene éxito, SessionID contiene un código de error con el StatusMsg que contiene el texto que describe el fallo [4].

Tipo de Comunicación	Generadores Tektronix AFG3000	Osciloscopio
LAN	Soportado a través del servidor VXI 11	Requiere TekVISA
GPIB	Requiere TekVISA	Requiere TekVISA
RS232	No soportado	Requiere TekVISA
USB	Requiere TekVISA	Requiere TekVISA

Tabla 4.1. Tipos de conexiones para generadores de funciones Tektronix AFG3000 [4].

4ºCrear una forma de onda con las funciones matemáticas de MATLAB como Seno, Exponencial, y otros:

La siguiente secuencia de comandos, por ejemplo, crea una forma de onda sinusoidal con un millar de puntos y la almacena en una variable llamada de Datos.

```
Frecuencia = 1e2;

Periodo= 1/Frecuencia;

PTS = [1:1000] * Periodo;

Datos = sin (PTS);
```

5º. Utilizar esta variable para la transferencia de la onda sinusoidal creada para transmitirla al generador, como se muestra:

```
[Status, StatusMsg] = SendWfm (SessionID, Datos, 'example.wfm ', 1000)
```

donde SessionID es la sesión que se ha abierto antes, example.wfm es el nombre del archivo de la forma de onda, y 1000 es el número de puntos que componen la forma de onda.

Si la transferencia de archivos se realiza correctamente, debería ver el archivo de forma de onda con el nombre seleccionado en el directorio actual del generador.

6º. Para cerrar la sesión se usa el comando CloseSession como se muestra:

```
CloseSession (<SessionID>)
```

donde SessionID es la sesión que ha creado.

4.4.2 Controlando el instrumento

Se puede controlar el instrumento desde el entorno de MATLAB. Para ello, hay que seguir estos pasos:

1. Abrir una sesión con el comando NewSession.

2. Una vez conectado, utilizar el comando Write para enviar un comando que es compatible con el generador.

3. Utilizar el comando Query para consultar los parámetros del instrumento.

Por ejemplo, el comando siguiente muestra cómo cargar un archivo y la salida desde el canal 1 del generador AWG:

```
[Status, StausMsg] = LoadWfm (SessionID, 'example.Wfm', 1)
```

Este comando intenta cargar el archivo "example.Wfm" de la ruta del directorio actual y la salida desde el canal 1 del generador. Este comando no funciona si el archivo es de un formato incorrecto o no se encuentran en el directorio actual.

No se podrá ver ninguna salida si el número de canal seleccionado no está disponible o no es compatible con el instrumento conectado.

Programa de ejemplo:

```
Un código de ejemplo muy preliminar para mostrar cómo
utilizar las funciones de MATLAB para comunicarse con un
generador desde MATLAB:
Este script sólo con cadenas de recursos Visa. Apague echo.
echo off
Abrir una sesión. Si se está conectando a través de LAN a
un generador el primer parámetro a NewSession sería la
dirección IP y el segundo parámetro sería la cadena
'tcpip'.
Así sería para GPIB:
s = NewSession ('gpib0:: 9:: instr', 'gpib');
Hay salidas de dos valores: Estado de la consulta, y de la
respuesta del instrumento.
[status, idn]= query (s, '* idn? ');

Escribir un solo comando para el generador conectado. El
status
será cero si el comando Write es exitoso
status = Write (s, 'Output1:State One ');
El siguiente código crea una forma de onda senoidal de
1000 puntos
Frecuencia = 1e2;
Periodo =1/Frecuencia;
Pts = [1:1000] * Periodo;
Data = sin (Pts);
El siguiente comando transfiere la forma de onda de la
variable Data al directorio actual del generador
TransferWfm (s, 'example.wfm', Data, 1000)
Cierra la conexión
CloseSession (s);
```

Comandos

ArbExpress implementa los siguientes comandos para la comunicación con el generador Tektronix:

Hay dos comandos para manejo de sesiones-NewSession y CloseSession y cinco comandos de control de instrumentos y de transferencia de datos: Read, Write, LoadWfm, TransferWfm, y Query.

Descripción del comando NewSession

Este comando abre una sesión de comunicación con un generador Tektronix conectado al PC a través de GPIB o LAN.

Sintaxis

```
[nSessionID, strErrMsg] = NewSession (strResourceID, strBusType)
```

Parámetros

strResourceID- Es un cadena de MATLAB que contiene el nombre del recurso en el formato estándar de Visa.

strBusType-Bus de comunicación utilizado para la conexión. Los valores admitidos son "GPIB" y "TCPIP".

Valores de retorno

nSessionID-Una variable numérica que contiene el identificador de sesión que se utilizará en las comunicaciones posteriores con el instrumento.

strErrMsg-Contiene la descripción de error si no consigue conectar con el instrumento. Cuando la conexión es exitosa que contiene una cadena nula.

Ejemplo:

Para GPIB: `[SessionID, StatusMsg] =NewSession ('GPIB0:: 9:: INSTR', 'GPIB)`

Para LAN: `[SessionID, StatusMsg] = NewSession ("TCPIP:: 10.20.30.40:: INSTR ',' TCPIP ')`

Para USB: `[SessionID, StatusMsg] = NewSession ('USB0:: 0x0699: 0x0343:: JU010107:: INSTR', 'USB')`

Descripción del comando CloseSession

Este comando cierra la conexión con el instrumento.

Sintaxis

```
CloseSession (nSessionID)
```

Parámetros

nSessionID- Es un valor numérico que la sesión obtiene cuando el comando de NewSession fue llamado.

Valores de retorno

Ninguno.

Nota: Se debe tener una conexión activa con un instrumento antes de llamar a CloseSession.

Ejemplo

```
CloseSession (<sessionID>)
```

Descripción del comando Write

Este comando envía un comando SCPI al instrumento conectado.

Sintaxis

```
[nStatus, strErrMsg] = Write (nSessionID, strCmd)
```

Parámetros

nSessionID- Sesión ID obtenida por el comando NewSession.

strCmd- Cadena del comando SCPI

Valores de retorno

nStatus- Cero si el código de éxito, error si la operación ha fallado.

Ejemplo

```
[NStatus, strErrMsg] = Write (nSessionID, "* IDN?")
```

Descripción del comando Read

Este comando lee las respuestas de los instrumentos. Responde a la consulta enviada en una llamada anterior por el comando Write o responde con un mensaje de error si la consulta está mal hecha.

Sintaxis

```
[NStatus, strResponse] = Read (nSessionID)
```

Parámetros

nSessionID- Sesión ID obtenida del comando NewSession

Valores de retorno

nStatus-Cero si el código de éxito, error si la operación ha fallado.

strResponse-Respuesta a la consulta enviada en el parámetro strQuery o responde con un mensaje de error si la consulta falla.

Nota: Read devolverá una respuesta adecuada sólo cuando es llamado inmediatamente después de una operación Write que escribió uno de los comandos de consulta SCPI.

Ejemplo

```
Read (nSessionID)
```

Descripción del comando LoadWfm

Este comando carga una forma de onda en la memoria interna del generador Tektronix conectado y lo muestra desde el canal seleccionado.

Sintaxis

```
[NStatus, strErrMsg] = LoadWfm (nSessionID, strWfmName, nCh)
```

Parámetros

nSessionID-Sesión ID obtenida por el comando NewSession llamado.

strWfmName-Nombre del archivo de forma de onda, con o sin la extensión.

nCh-Número de canal.

Valor de retorno

El comando devuelve siempre cero como valor de retorno.

Nota: El archivo de forma de onda que se está tratando de carga debe estar en el directorio actual. Si el archivo está en cualquier otro directorio, se debe cambiar la ruta al directorio donde se encuentra el archivo deseado mediante el uso de MMEM: comando REDC. Cuando el comando de carga es enviada, el instrumento toma algún tiempo para cargar el archivo en especial si se trata de archivos grandes. Dado que la llamada es sincrónica, el programa no volverá hasta que la operación de carga se ha completado en el instrumento.

También hay que tener en cuenta que si el archivo de la forma de onda está dañado o en el formato incorrecto, el instrumento no lo cargará.

Ejemplo

```
[Status, StatusMsg] = LoadWfm (SessionID, 'example.Wfm', 1)
```

Descripción del comando TransferWfm

Este comando transfiere una forma de onda desde el entorno de MATLAB a la memoria del instrumento conectado.

Sintaxis

```
TransferWfm [nSessionID, strWfmName, double dDataPoints [], nSize)
```

Parámetros

nSessionID- Sesión ID obtenida por el comando NewSession llamado.

strWfmName-El nombre del archivo de forma de onda que se creará en el instrumento.

dDataPoints-Matriz unidimensional de doubles que contienen los puntos de datos.

nSize-Número de puntos de datos en el archivo.

Valores de retorno

Ninguno.

Nota: Esta función se usa para transferir una matriz de valores de datos a un generador y la almacena en forma de un archivo de forma de onda. Hay que ser cauteloso al usar esta función, porque si se elige un nombre de archivo que ya existe en el directorio actual del instrumento, será reemplazado sin previo aviso.

También hay que tener en cuenta que esto es sólo un comando de transferencia y al transferir el archivo no lo está cargando en la memoria interna del instrumento.

Ejemplo:

```
TransferWfm (SessionID, 'example.wfm', Data, 1000)
```

Descripción del comando Query

Este comando envía una consulta y devuelve la respuesta del instrumento.

Sintaxis

```
[nStatus, strResponse] = Query(nSessionID, strQuery)
```

Parámetros

nSessionID- Sesión ID obtenida por el comando NewSession llamado.

strQuery- Consulta estándar SCPI

Valores de retorno

nStatus-Cero si tiene éxito, el código de error de lo contrario

strResponse-Respuesta a la consulta enviada en el parámetro strQuery o mensaje de error si la consulta ha fallado.

Nota: Esta es una combinación de comandos de Write y Read. El valor de retorno siempre devuelve el resultado de la operación de lectura.

Ejemplo:

```
[nStatus, strResponse] = Query (nSessionID, '* IDN?')
```

Capítulo 5

Interfaces de usuario con Matlab

5.1 Introducción a Matlab

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices"), es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.
El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, que son Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI).
Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (*toolboxes*); y las de Simulink con los paquetes de bloques (*blocksets*).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

Fue creado por Cleve Moler en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje.
El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices *LINPACK* y *EISPACK* sin tener que usar Fortran.

En 2004, se estimaba que MATLAB era empleado por más de un millón de personas en ámbitos académicos y empresariales [6].

MATLAB es un programa de cálculo numérico orientado a matrices. Por tanto, será más eficiente si se diseñan los algoritmos en términos de matrices y vectores.

Ejemplo "Hola Mundo":
Éste es el tradicional programa Hola Mundo hecho con el lenguaje de MATLAB:

```
>> disp('Hola Mundo');        % Muestra el mensaje
Hola Mundo
```

Cajas de herramientas y paquetes de bloques

Las funcionalidades de Matlab se agrupan en más de 35 cajas de herramientas y paquetes de bloques (para Simulink) clasificadas en las categorías mostradas en la siguiente figura [6].

MATLAB (Cajas de herramientas)	Simulink
Matemáticas y Optimización	Modelado de punto fijo
Estadística y Análisis de datos	Modelado basado en eventos
Diseño de sistemas de control y análisis	Modelado físico
Procesado de señal y comunicaciones	Gráficos de simulación
Procesado de imagen	Diseño de sistemas de control y análisis
Pruebas y medidas	Procesado de señal y comunicaciones
Biología computacional	Generación de código
Modelado y análisis financiero	Prototipos de control rápido y SW/HW HIL
Desarrollo de aplicaciones	Tarjetas integradas
Informes y conexión a bases de datos	Verificación, validación y comprobación

Tabla 5.1. Funcionalidades de MATLAB [6].

Durante mucho tiempo hubo críticas porque MATLAB es un producto propietario de The Mathworks, y los usuarios están sujetos y bloqueados al vendedor. Recientemente se ha proporcionado una herramienta adicional llamada MATLAB Builder bajo la sección de herramientas "Application Deployment" para utilizar funciones MATLAB como archivos de biblioteca que pueden ser usados con ambientes de construcción de aplicación .NET o Java. Pero la desventaja es que el computador donde la aplicación tiene que ser utilizada necesita MCR (MATLAB Component Runtime) para que los archivos MATLAB funcionen correctamente. MCR se puede distribuir libremente con los archivos de biblioteca generados por el compilador MATLAB [6].

5.2 GUIDES de Matlab

Matlab GUIDE es un entorno de programación visual disponible en Matlab para realizar y ejecutar programas que necesiten ingreso continuo de datos.
Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.

El lenguaje más habitual para crear GUI-s es Java, ya que tiene la enorme ventaja de funcionar en cualquier máquina, sin embargo Java resulta muy lento para hacer cálculos eficientemente, y es aquí donde Matlab es más poderoso. Por otro lado, las GUI-s creadas con Matlab pueden ser entregadas al ordenador de un cliente (quien posiblemente no tenga más que un navegador) y ser ejecutadas en el ordenador de quien creó la interfaz en Matlab (y que por supuesto tiene un Matlab funcionando), de modo que la ventaja relativa de Java está parcialmente ofertada también por Matlab.

Las GUI-s son herramientas muy útiles para entregar aplicaciones a aquellas personas que no saben lo suficiente de programación y que quieren beneficiarse de las ventajas de un programa.

Para iniciar este proyecto, lo podemos hacer de dos maneras:
• Ejecutando la siguiente instrucción en la ventana de comandos:

```
>> guide
```

• Haciendo un click en el ícono que muestra la figura [7]:

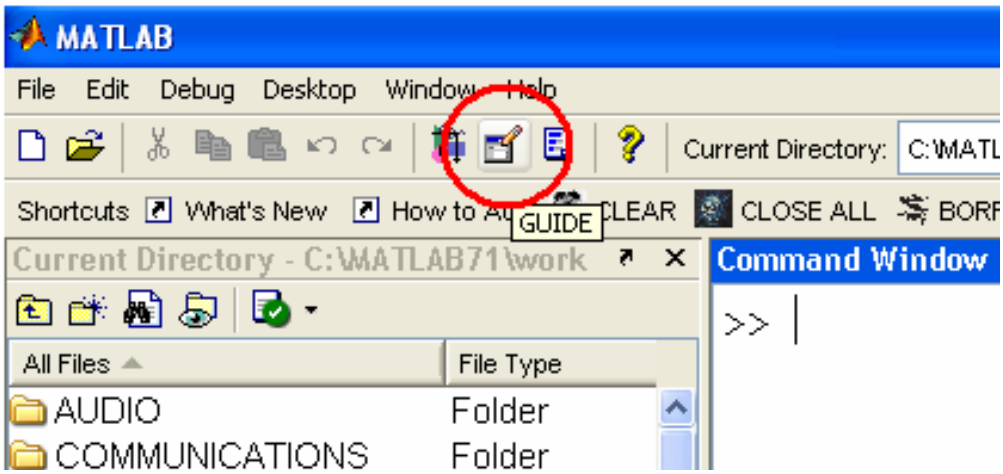


Figura 5.1. Icono GUIDE.

Se presenta el siguiente cuadro de diálogo:

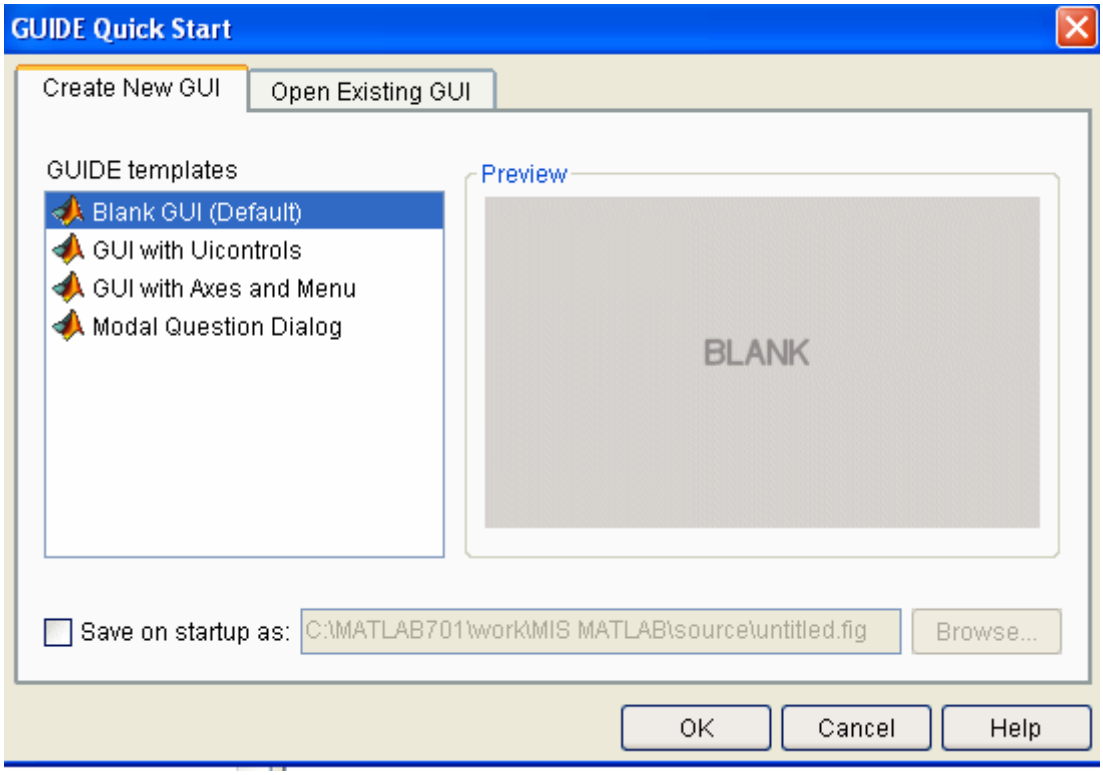


Figura 5.2. Ventana de inicio GUIDE.

Se presentan las siguientes opciones:

a) Blank GUI (Default)
La opción de interfaz gráfica de usuario en blanco (viene predeterminada), presenta un formulario nuevo, en el cual se puede diseñar un programa.

b) GUI with Uicontrols
Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en alguno de los dos sistemas de unidades. Podemos ejecutar este ejemplo y obtener resultados.

c) GUI with Axes and Menu
Esta opción es otro ejemplo el cual contiene el menú File con las opciones Open, Print y Close. En el formulario tiene un *Popup menu*, un *push button* y un objeto *Axes*, podemos ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú despegable y haciendo clic en el botón de comando.

d) Modal Question Dialog
Con esta opción se muestra en la pantalla un cuadro de diálogo común, el cual consta de una pequeña imagen, una etiqueta y dos botones *Yes* y *No*, dependiendo del botón que se presione, el GUI retorna el texto seleccionado (la cadena de caracteres 'Yes' o 'No').

Se suele elegir la primera opción, *Blank GUI,* y tenemos:

Figura 5.3. Entorno de diseño GUI [7].

La interfaz gráfica cuenta con las siguientes herramientas:

	Alinear objetos.
	Editor de menú.
	Editor de orden de etiqueta.
	Editor del M-file.
	Propiedades de objetos.
	Navegador de objetos.
	Grabar y ejecutar (ctrl. + T).

Tabla 5.2. Herramientas de la interfaz gráfica.

Para obtener la etiqueta de cada elemento de la paleta de componentes ejecutamos: *File>>Preferentes* y seleccionamos *Show names in component palette.* Tenemos la siguiente presentación:

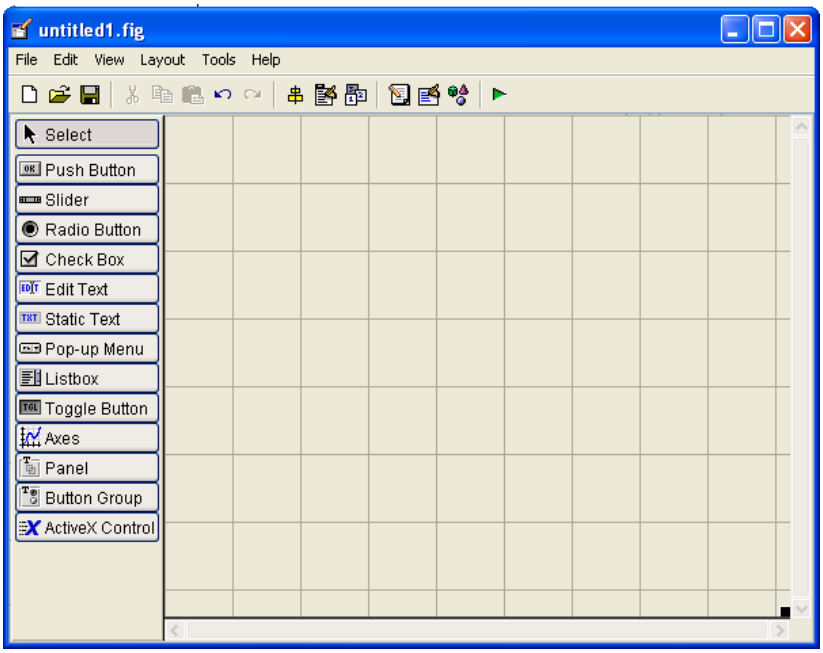


Figura 5.4. Entorno de diseño GUI. Componentes etiquetadas.

5.3 Componentes de las Guides

La siguiente tabla muestra una descripción de los componentes:

Control	*Valor de estilo*	*Descripción*
Check box	'checkbox'	Indica el estado de una opción o atributo
Editable Text	'edit'	Caja para editar texto
Pop-up menu	'popupmenu'	Provee una lista de opciones
List Box	'listbox'	Muestra una lista deslizable
Push Button	'pushbutton'	Invoca un evento inmediatamente
Radio Button	'radio'	Indica una opción que puede ser seleccionada
Toggle Button	'togglebutton'	Solo dos estados, "on" o "off"
Slider	'slider'	Usado para representar un rango de valores
Static Text	'text'	Muestra un string de texto en una caja
Panel button		Agrupa botones como un grupo
Button Group		Permite exclusividad de selección con los radio button

Tabla 5.3 Descripción de los componentes [7].

5.3.1 PROPIEDADES DE LOS COMPONENTES

Cada uno de los elementos de GUI, tiene un conjunto de opciones que podemos acceder con clic derecho.

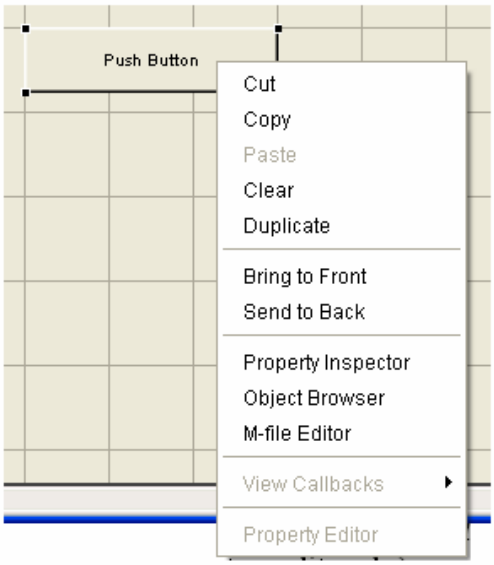


Figura 5.5 Opciones del componente.

La opción *Property Inspector* nos permite personalizar cada elemento.

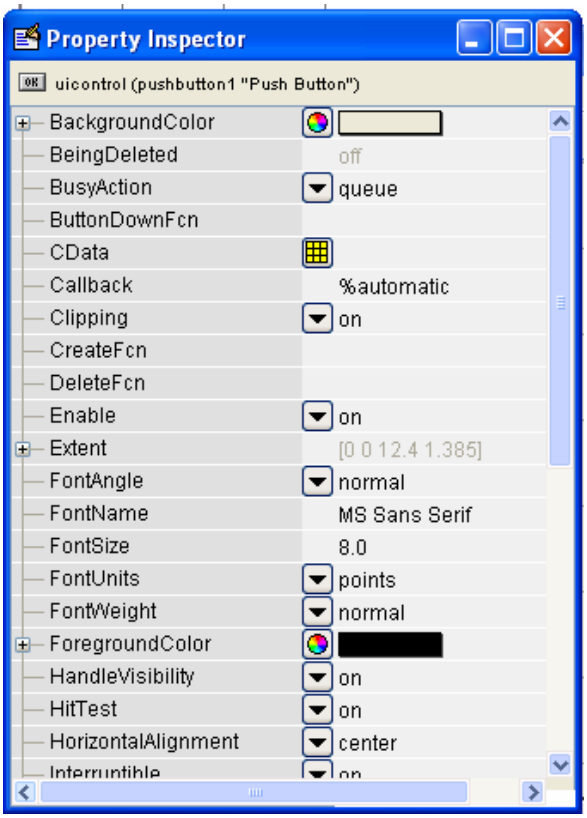


Figura 5.6 Entorno Property Inspector.

Al hacer clic derecho en el elemento ubicado en el área de diseño, una de las opciones más importantes es ***View Callbacks***, la cual, al ejecutarla, abre el archivo *.m* asociado a nuestro diseño y nos posiciona en la parte del programa que corresponde a la subrutina que se ejecutará cuando se realice una determinada acción sobre el elemento que estamos editando.

Por ejemplo, al ejecutar *View Callbacks>>Callbacks* en el *Push Button*, nos ubicaremos en la parte del programa:

```
function pushbutton1_Callback(hObject, eventdata, handles)
%hObject handle to pushbutton1 (see GCBO)
%eventdata reserved-to be defined in a future version of MATLAB
%handles structure with handles and user data (see GUIDATA)
```

5.4 Programación de las Guides

5.4.1 FUNCIONAMIENTO DE UNA APLICACIÓN GUI

Una aplicación GUIDE consta de dos archivos: *.m* y *.fig*. El archivo *.m* es el que contiene el código con las correspondencias de los botones de control de la interfaz y el archivo *.fig* contiene los elementos gráficos.

Cada vez que se adicione un nuevo elemento en la interfaz gráfica, se genera automáticamente código en el archivo *.m*.

Para ejecutar una Interfaz Gráfica, si la hemos etiquetado con el nombre *curso.fig*, simplemente ejecutamos en la ventana de comandos *>> curso*. O haciendo clic derecho en el m-file y seleccionando la opción *RUN*.

5.4.2 MANEJO DE DATOS ENTRE LOS ELEMENTOS DE LA APLICACIÓN Y EL ARCHIVO .M

Todos los valores de las propiedades de los elementos (color, valor, posición, string…) y los valores de las variables transitorias del programa se almacenan en una estructura, los cuales son accedidos mediante un único y mismo *identificador* para todos éstos. Tomando el programa listado anteriormente, el identificador se asigna en:

```
handles.output = hObject;
```

handles, es nuestro identificador a los datos de la aplicación. Esta definición de identificador es salvada con la siguiente instrucción:

```
guidata(hObject, handles);
```

guidata, es la sentencia para salvar los datos de la aplicación.

Aviso: *guidata* es la función que guarda las variables y propiedades de los elementos en la estructura de datos de la aplicación, por lo tanto, como regla general, en cada subrutina se debe escribir en la última línea lo siguiente:

```
guidata(hObject,handles);
```

Esta sentencia nos garantiza que cualquier cambio o asignación de propiedades o variables quede almacenado.
Por ejemplo, si dentro de una subrutina una operación dio como resultado una variable *diego* para poder utilizarla desde el programa u otra subrutina debemos salvarla de la siguiente manera:

```
handles.diego=diego;
guidata(hObject,handles);
```

La primera línea crea la variable *diego* a la estructura de datos de la aplicación apuntada por *handles* y la segunda graba el valor.

5.4.3 SENTENCIAS GET Y SET

La asignación u obtención de valores de los componentes se realiza mediante las sentencias *get* y *set*. Por ejemplo si queremos que la variable *utpl* tenga el valor del *Slider* escribimos:

```
utpl= get(handles.slider1,'Value');
```

Notar que siempre se obtienen los datos a través de los identificadores *handles*.
Para asignar el valor a la variable *utpl* al *statictext* etiquetada como *text1* escribimos:

```
set(handles.text1,'String',utpl);%Escribe el valor del Slider...
%en static-text
```

5.4.4 EJEMPLO DE UNA APLICACIÓN

La siguiente figura muestra un ejemplo de aplicación, más adelante se muestra otro ejemplo (ruido gaussiano) con su correspondiente archivo.m.

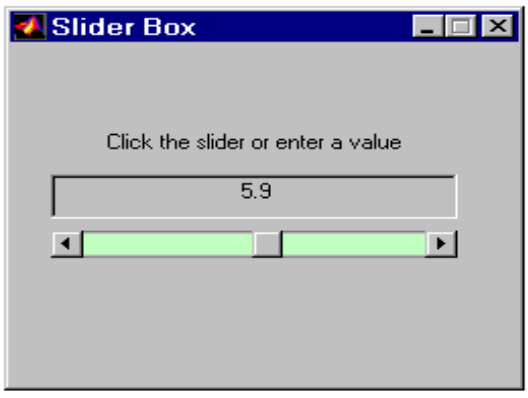


Figura 5.7 Ejemplo de una aplicación.

Ejecución: Desde la ventana de comando del Matlab se debe ejecutar el comando *guide*.

Esto abre la consola de edición de la parte gráfica de la aplicación a implementar (.fig), es decir, colocar botones, cuadros de dialogo, graficas, texto, etc.

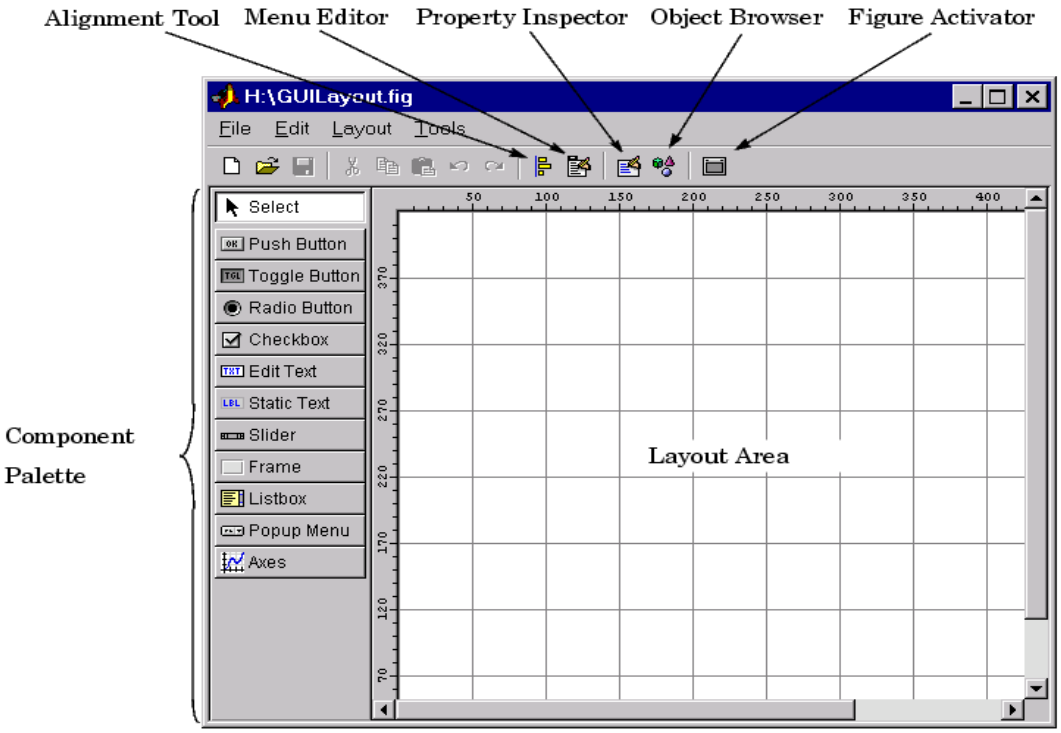


Figura 5.8 Consola de edición de la parte gráfica.

Cada uno de estos elementos tienen un conjunto de propiedades a las cuales podemos acceder con el botón derecho del ratón, una vez clicado este aparece el siguiente cuadro:

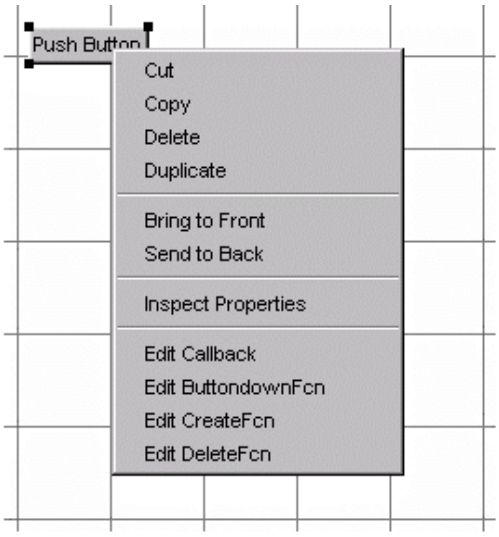


Figura 5.9 Conjunto de propiedades.

Para editar las propiedades de cada elemento seleccionamos la opción *Properties Inspector* y se abre una consola (la cual variará según que elemento se este editando) con todas las propiedades que podemos editar, ejemplo: color, posición, tamaño, fuente, etc.

Una de las opciones de mayor interés en la figura anterior es *Edit Callback*. Esta última abre el archivo .m asociado (ejecutable Matlab) y nos posiciona en la sección del programa que corresponde a la subrutina que se ejecutara cuando se realice una determinada acción sobre el elemento que se está editando.

Por ejemplo para el botón 1, Edit Callback nos posiciona en la siguiente parte del programa:

```
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)

% Stub for Callback of the uicontrol handles.pushbutton1.

disp('pushbutton1 Callback not implemented yet.')
```

Una aplicación Guide consta de dos archivos uno .m ejecutable y otro .fig la parte gráfica. Las dos partes están unidas a través de las subrutinas *callback*. Una vez que se graba los archivos desde la consola de emisión (si salvamos la .fig automáticamente lo hace el .m asociado) podemos ejecutar el programa en la ventana de comando de Matlab solamente escribiendo el nombre del archivo.

Por ejemplo si guardamos un archivo ej.fig y ej.m escribiendo *ej* y presionando *enter* se ejecuta el programa.

El archivo .m que se crea tiene una estructura predeterminada. Consta de un encabezado y a continuación viene el código correspondiente a las siguientes subrutinas. Por ejemplo una aplicación cuya figura tenga tres botones, un grafico y un cuadro de edición tendrá un archivo .m con la siguiente estructura inicial como la siguiente:

```
function varargout = untitled1(varargin)
% UNTITLED1 Application M-file for untitled1.fig
% FIG = UNTITLED1 launch untitled1 GUI.
% UNTITLED1('callback_name', ...) invoke the named callback.
% Last Modified by GUIDE v2.0 20-Aug-2002 19:57:33
if nargin == 0  % LAUNCH GUI
      fig = openfig(mfilename,'reuse');
      % Use system color scheme for figure:
      set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
      % Generate a structure of handles to pass to callbacks, and
store it.
      handles = guihandles(fig);
      guidata(fig, handles);
      if nargout > 0
            varargout{1} = fig;
      end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
      try
            [varargout{1:nargout}] = feval(varargin{:});
            % FEVAL switchyard
      catch
            disp(lasterr);
      end
end
```

Encabezado

handles, Puntero a todas variables y elementos de la aplicación

guidata(), comando para guardar las variables de la aplicación

```
function varargout = pushbutton1_Callback(h, eventdata, handles,
varargin)

% Stub for Callback of the uicontrol handles.pushbutton1.

disp('pushbutton1 Callback not implemented yet.')

% ------------------------------------------------------------------

function varargout = pushbutton2_Callback(h, eventdata, handles,
varargin)

% Stub for Callback of the uicontrol handles.pushbutton2.

disp('pushbutton2 Callback not implemented yet.')

% ------------------------------------------------------------------

function varargout = pushbutton3_Callback(h, eventdata, handles,
varargin)

% Stub for Callback of the uicontrol handles.pushbutton3.

disp('pushbutton3 Callback not implemented yet.')

% ------------------------------------------------------------------

function varargout = edit1_Callback(h, eventdata, handles, varargin)

% Stub for Callback of the uicontrol handles.edit1.

disp('edit1 Callback not implemented yet.')
```

Subrutina Botón 1

Subrutina Botón 2

Subrutina Botón 3

Subrutina Cuadro de Edición 1

La siguiente figura muestra la interacción entre la figura (.fig) y el archivo (.m):

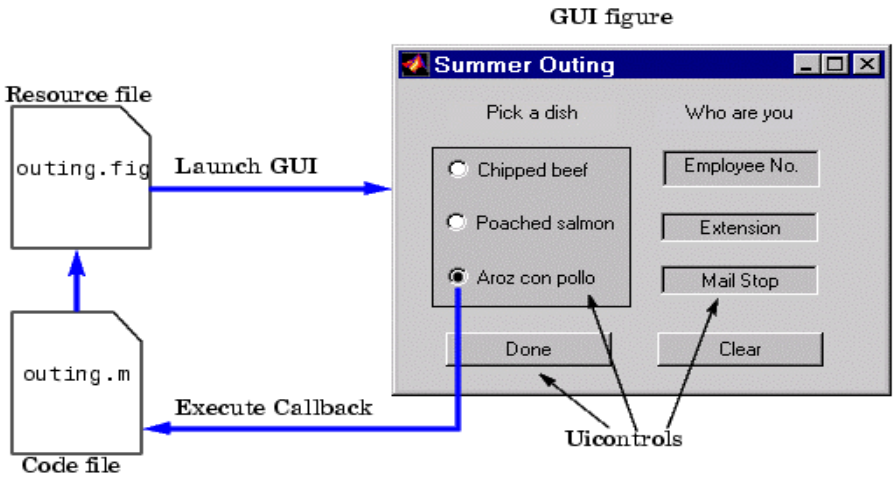


Figura 5.10 Interacción entre la figura (.fig) y el archivo (.m).

***Observación* sobre Salvar**: Si se guarda una modificación del archivo cuando aparece el cuadro de dialogo (ver figura) que pregunta si hacemos Replace o Append debemos elegir Append, si se selecciona Replace borra todo el archivo .m es decir lo pone a cero, no utilizar Replace.

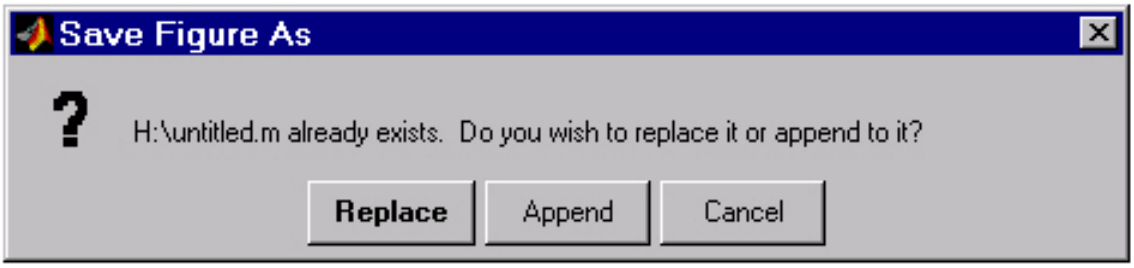


Figura 5.11 Cuadro de dialogo que pregunta si remplazamos.

5.4.4.1 Configuración Estándar de la Aplicación

Sobre las propiedades del elemento Figure:

Application Options: Al seleccionar esta opción se abrirá la siguiente ventana:

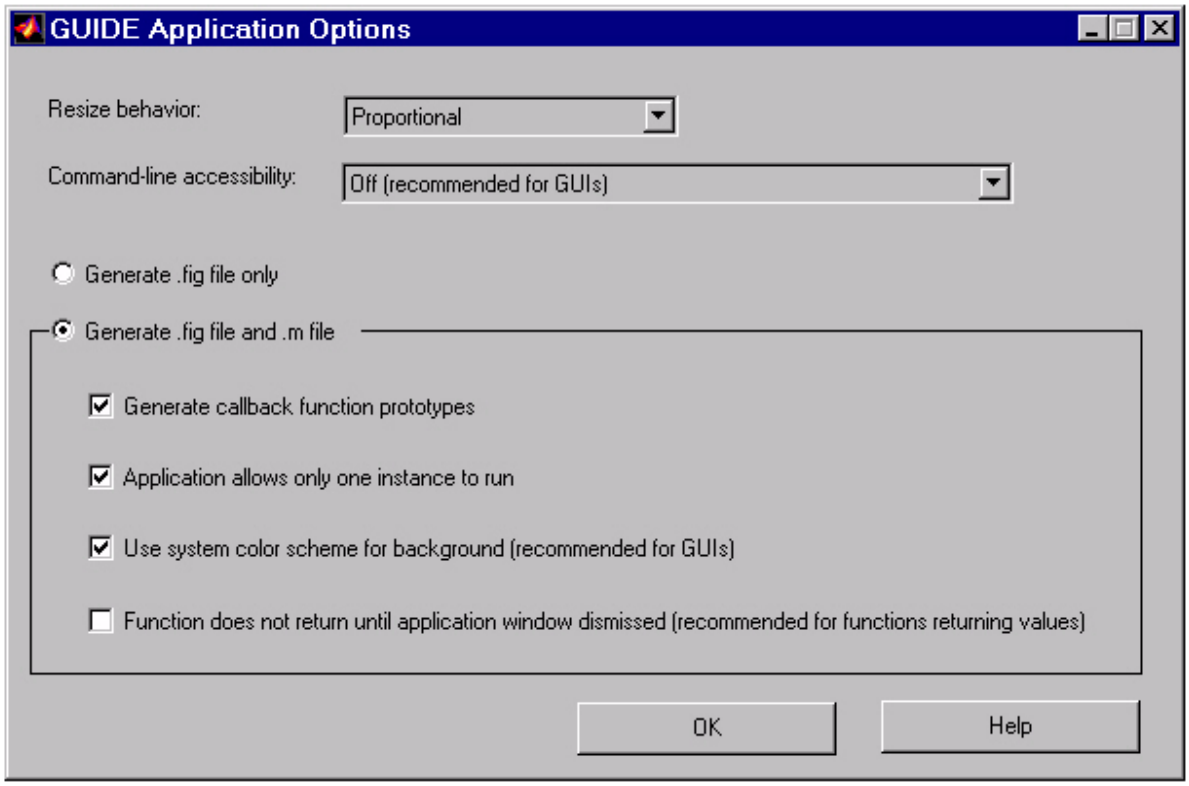


Figura 5.12 Propiedades del elemento Figure. *Application Options.*

5.4.4.2 Manejo de datos entre los elementos de la aplicación y el archivo .m

Todos los valores de las propiedades de los elementos (ejemplo: color, valor, posición, string, etc.) y los valores de las variables transitorias del programa se guardan en una estructura (structura matlab: puede almacenar matrices, string, arreglos, vectores, etc.) los cuales son accedidos mediante un único y mismo puntero para todos estos. El nombre del puntero se asigna en el encabezado del archivo .m, tomando por ejemplo el programa listado anteriormente el puntero se asigna en:

```
handles = guihandles(fig);
```

handles , es entonces el puntero a los datos de la aplicación .

Esta definición de puntero es salvada con la siguiente instrucción

```
guidata(fig, handles);
```

guidata, es entonces la función para salvar los datos de la aplicación.

Importante: guidata es la función que guarda las variables y propiedades de los elementos en la estructura de datos de la aplicación, entonces como regla general en cada subrutina se debe escribir en la última línea lo siguiente:

```
guidata(gcbo,handles);
```

Esto nos garantiza que cualquier cambio o asignación de propiedades o variables quede salvado.

Por ejemplo si dentro de una subrutina una operación dio como resultado una variable *fi* para poder utilizarla desde el programa u otra subrutina debemos salvarla de la siguiente manera:

```
handles.fi=fi;

guidata(gcbo,handles);
```

La primera línea crea la variable *fi* a la estructura de datos de la aplicación apuntada por *handles* y la segunda graba el valor.

5.4.4.3 Get y Set

La transferencia u obtención de los valores de las propiedades de los elementos se realiza mediante las funciones get y set. Por ejemplo si queremos que la variable *fi* tenga el valor del slider escribimos:

```
fi=get(handles.slider,'value');
```

Observar que siempre se obtienen los datos a través del puntero *handles*.

Ahora si quisiéramos hacer al revés y asignarle el valor la variable *fi* al slider debemos escribir:

```
set(handles.slider,'value',fi);
```

Para aclarar todos los conceptos anteriores se muestra una aplicación de ejemplo cuya presentación y archivo .m se listan a continuación. Lo que realiza este programa es dibujar un ruido Gaussiano (350 muestras) cuya varianza se elige con la barra de desplazamiento, el valor de esta se muestra en un cuadro de edición y luego con un botón denominado *plot* se dibuja.

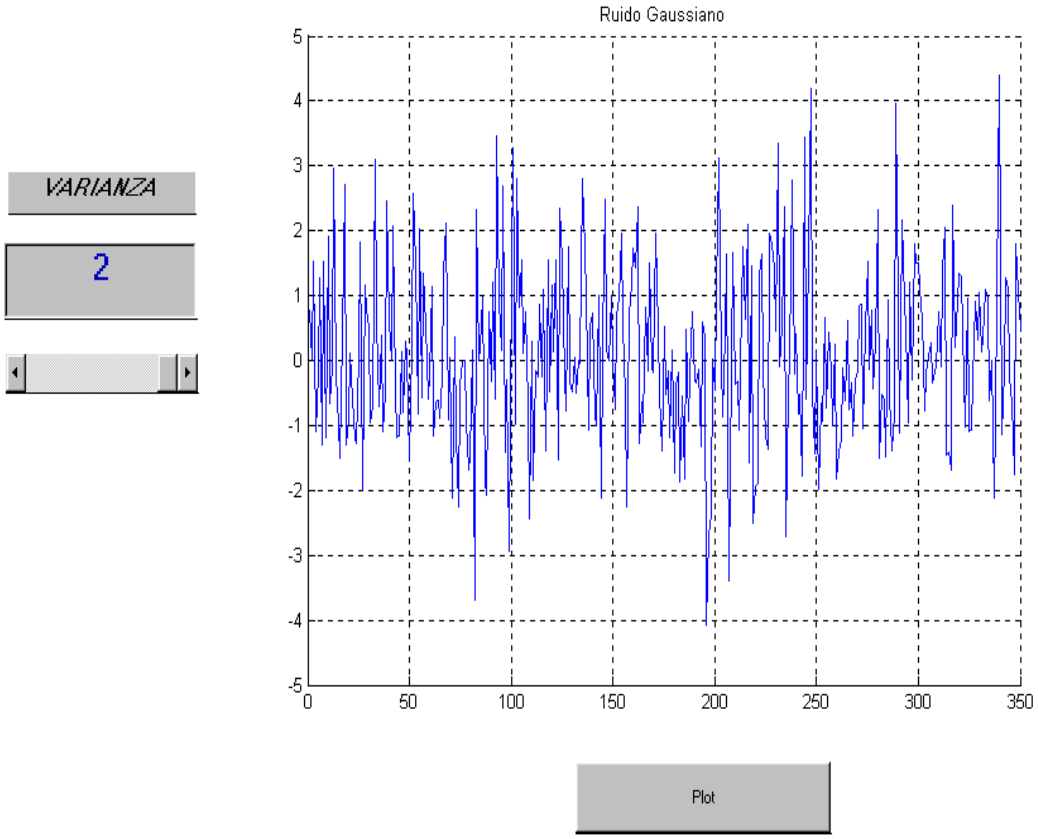


Figura 5.13 Aplicación de ejemplo. Programa que dibuja un ruido Gaussiano.

```
function varargout = tut(varargin)
% TUT Application M-file for tut.fig
%    FIG = TUT launch tut GUI.
%    TUT('callback_name', ...) invoke the named callba
if nargin == 0  % LAUNCH GUI
      fig = openfig(mfilename,'reuse');
      set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
      % Generate a structure of handles to pass to callbacks, and
store it.
      handles = guihandles(fig);
      guidata(fig, handles);
      if nargout > 0
            varargout{1} = fig;
      end
elseif ischar(varargin{1})  % INVOKE NAMED SUBFUNCTION OR CALLBACK
      try
            [varargout{1:nargout}] = feval(varargin{:});  % FEVAL
switchyard
      catch
            disp(lasterr);
      end
end
```

Inicialización, aquí se pueden añadir otros valores de inicialización

```
function varargout = pushbutton1_Callback(h, eventdata, handles,
varargin)

var=handles.var;

noise=randn(1,350);

noise=noise*sqrt(var);

%-----------------------------------------------------------

%instrucciones para imprimir en la grafica del programa y que no se
habra una ventana nueva

h=gca;

axes(h);

%instrucciones para imprimir en la grafica del programa y que no se
habra una ventana nueva

%esto es valido cuando solo tenemos una grafica, por ahora esta bien

%-----------------------------------------------------------

plot(noise);

title('Ruido Gaussiano');

grid on%

guidata(gcbo,handles);

% ------------------------------------------------------------

function varargout = edit1_Callback(h, eventdata, handles, varargin)

% ------------------------------------------------------------

function varargout = slider1_Callback(h, eventdata, handles, varargin)

var=get(handles.slider1,'value');

var=var*2;

set(handles.edit1,'string',num2str(var));

handles.var=var;

guidata(gcbo,handles);

%-------------------------------------------------------------
```

subrutina boton plot

subrutina slider

CAPÍTULO 6

La Interfaz Gráfica Generador de Funciones Arbitrarias

6.1. Introducción: proceso de diseño de una GUI

En primer lugar, antes de empezar a programar es imprescindible hablar con el usuario final de la GUI. Es importantísimo entender cuáles son las necesidades exactas que tienen que ser cubiertas por la aplicación. Para ello es necesario entender el tipo de datos y variables que son introducidas por el usuario, así como las excepciones que puedan producirse, los casos que ocurren pocas veces pero que hay que tener en cuenta, etc. También es necesario saber cómo quiere el usuario que se presenten los datos; si se necesitan gráficos o tablas que salgan por impresora, o cómo se guardan los resultados, dónde se guardan y en qué formato lo hacen. La parte del diseño es, con mucha diferencia, la más importante desde el punto de vista del usuario y por tanto también lo es desde el punto de vista empresarial. Para diseñar correctamente una GUI, lo mejor es hacerlo con papel y lápiz.

Desde el punto de vista empresarial, primero se presentaría un boceto al cliente y mejorarlo con él es la mejor opción. De esta manera se consigue que no haya sorpresas y evita que después de haber realizado un montón de trabajo luego haya que tirarlo a la basura y que encarezca mucho los proyectos, y además se consigue que el cliente se implique en el proyecto poniendo su talento y sus preferencias en la herramienta que al final usará él mismo.

Las GUI-s tienen que hacerse de modo que los botones estén donde la gente espera que estén. Si nuestra GUI tiene varias páginas distintas y en cada una de ellas hay un botón que dice “Guardar” es conveniente que ese botón esté localizado en el mismo sitio siempre. Todo esto parece ser de un sentido tan común que parece innecesario hacer notar que el papel y el lápiz son la mejor herramienta, sin embargo al hacer GUI-s sólo el sentido común tiene algún sentido.

Una vez que tenemos claro qué objetos tendrá la GUI, gráficos, textos, radio buttons, check boxes, edición de texto, entrada de valores, lectura de matrices, etc, y una vez que tengamos claro de qué forma aparecerán en la interfaz (el *layout*) es necesario hacer un programa de tipo *script* que tenga la misma funcionalidad que la GUI que queremos programar. Antes de incorporar el programa a la GUI, es necesario hacer todo tipo de pruebas con él hasta estar completamente seguros de que el programa que vamos a incorporar en la GUI es el programa que queremos. Para hacer las necesarias pruebas lo mejor es hacerlas sobre un *script* y no directamente sobre la GUI. Una vez que tengamos el *script* guardado podremos incorporar los distintos trozos del *script* en la GUI, de modo que al hacer las

pruebas sobre la GUI podamos contrastar los resultados con los que obtenemos del *script*.
Una vez hayamos acabado con los tests sobre la GUI definitiva y estemos completamente seguros de su correcto funcionamiento, la GUI puede ser entregada al usuario o en el caso empresarial al cliente.

6.2. Descripción de las ventanas que constituyen la aplicación Generador de Funciones Arbitrarias

6.2.1. Ventana principal y paneles que lo componen

La ventana principal del programa representa la integración de todas las funciones que han sido creadas para implementar este programa en Matlab en una única interfaz gráfica mostrada a continuación:

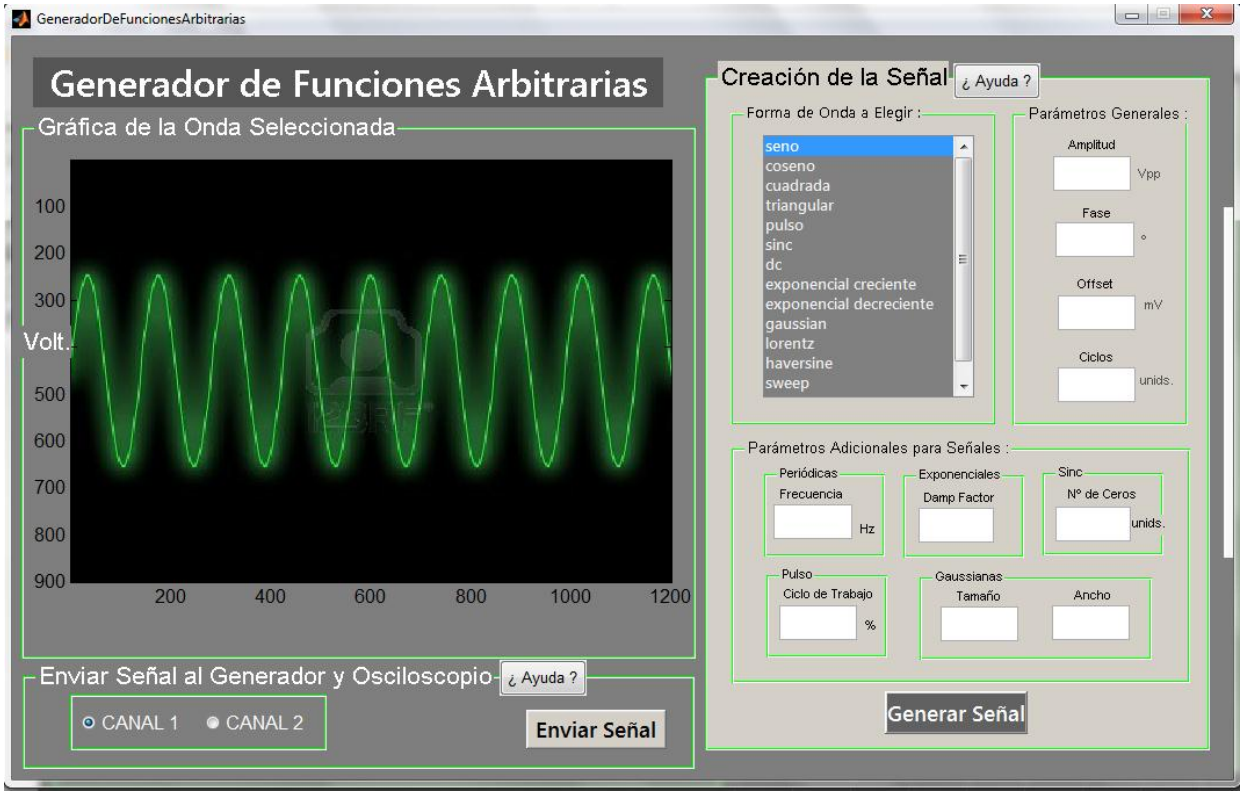


Figura 6.1. Ventana principal de la Interfaz Gráfica Diseñada

Esta interfaz gráfica consta de tres paneles principales:

Panel de Creación de la Señal, panel Grafica de la onda Seleccionada y panel Enviar señal al Generador y Osciloscopio.

Panel de Creación de la Señal:

Este panel es fundamental ya que en él el usuario se encargar de diseñar la señal, primeramente elegirá el tipo de onda a transmitir, y después introducirá todos los parámetros que necesita dicha señal.

Este panel se subdivide a su vez en otros tres paneles que son:

Panel Formas de Onda a elegir, panel de Parámetros generales y Panel de Parámetros adicionales, con estos tres paneles quedará la señal perfectamente definida, y lista para ser graficada y si se desea también puede ser enviada al generador de funciones Tektronix.

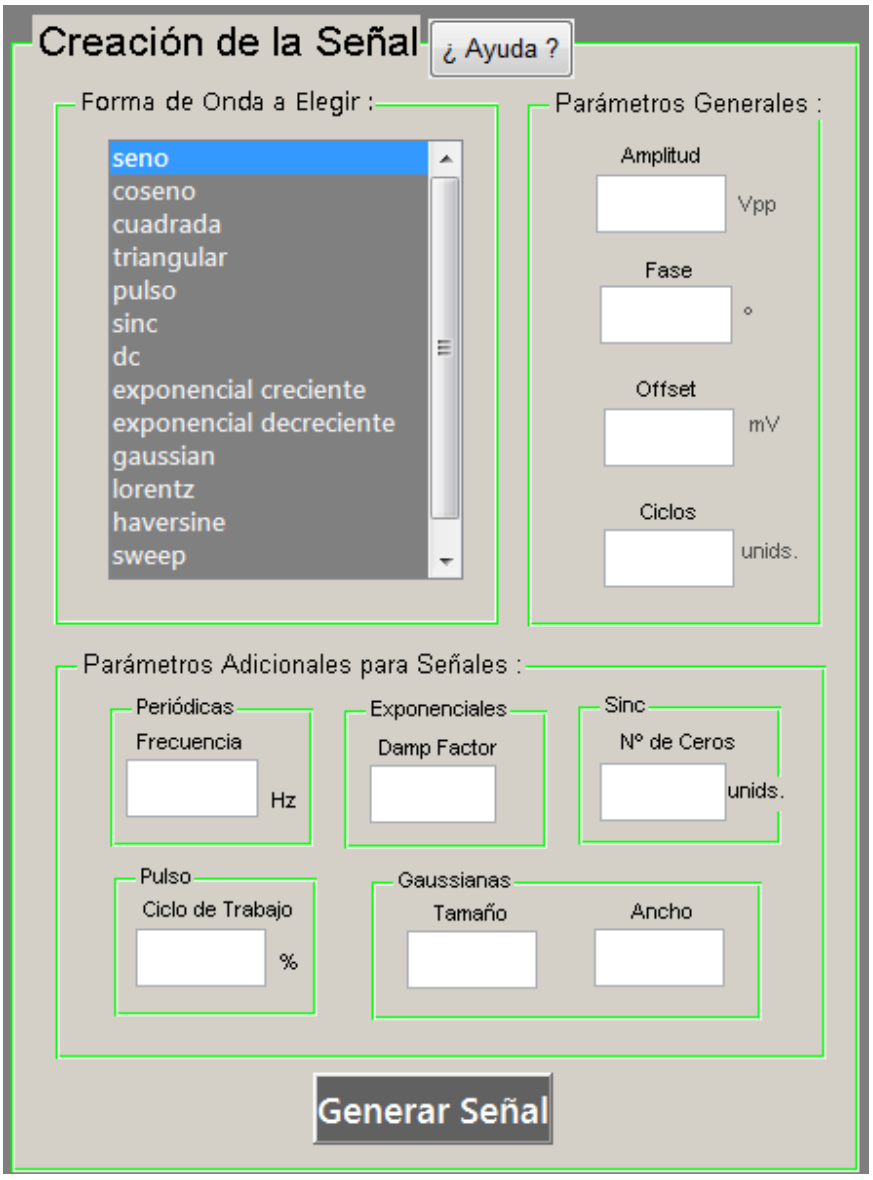


Figura 6.2. Panel de Creación de la señal

Subpaneles del Panel de Creación de la Señal

Panel Formas de Onda a elegir:

Aquí el usuario seleccionará la señal que desea observar, entre ellas se encuentran: Seno, coseno, cuadrada, triangular, pulso, sinc, dc, exponencial creciente y decreciente, señal gaussiana, lorentz y haversine.

Cada una de estas señales tiene una función que ha sido creada para este proyecto en Matlab de forma que el usuario pueda introducir y modificar los parámetros que desee, estas funciones están integradas dentro de la interfaz gráfica de modo que al seleccionar cada una de estas señales se hará una llamada a su función correspondiente.

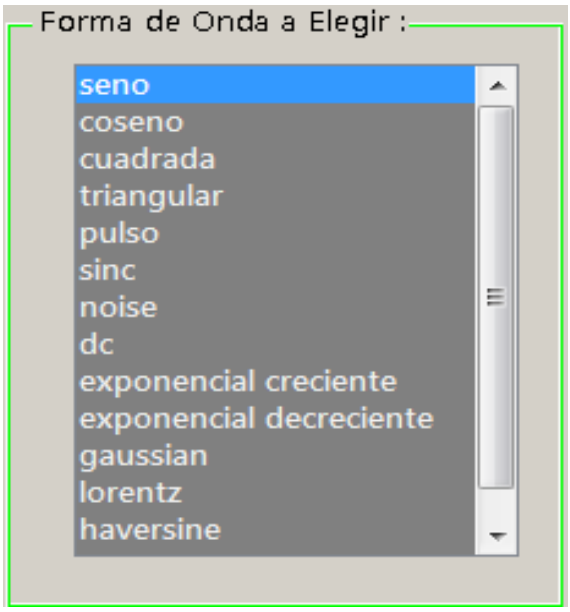


Figura 6.3. Panel de la Forma de Onda a Elegir

Panel de Parámetros generales:

Este panel está compuesto por los parámetros convencionales que componen una señal como son la Amplitud, la Fase, el Offset, y Ciclos que es el número de periodos que el usuario desea ver de la señal a representar.

Cada uno de estos parámetros han sido modificados dentro de los códigos correspondientes a cada función, de forma que estos parámetros se puedan introducir en la interfaz gráfica con las mismas unidades de magnitud que los generadores de funciones reales, es decir, la amplitud en Vpp (Voltios pico a pico), la fase en grados y el offset en mV (mili voltios).

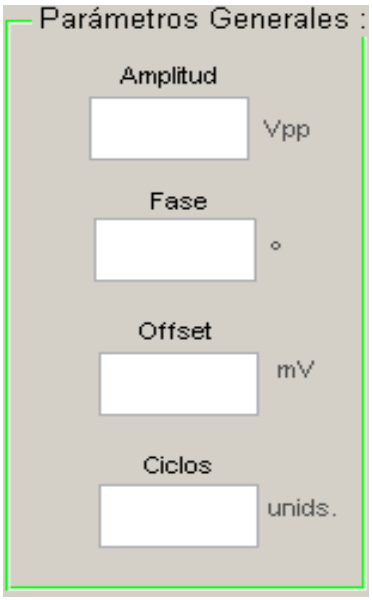


Figura 6.4. Panel de Parámetros generales a Introducir

Panel de Parámetros Adicionales:

Este panel se compone de los parámetros identificativos de cada señal, es decir de la frecuencia para las señales periódicas, el Damp Factor (Factor de Crecimiento) para las señales exponenciales, el número de cruces por cero para la Sinc, el Ciclo de Trabajo (Duty Cicle) para el pulso, y para las señales gaussianas el tamaño y el ancho de la señal, el tamaño también se usará para la señales Lorentz y Haversine.

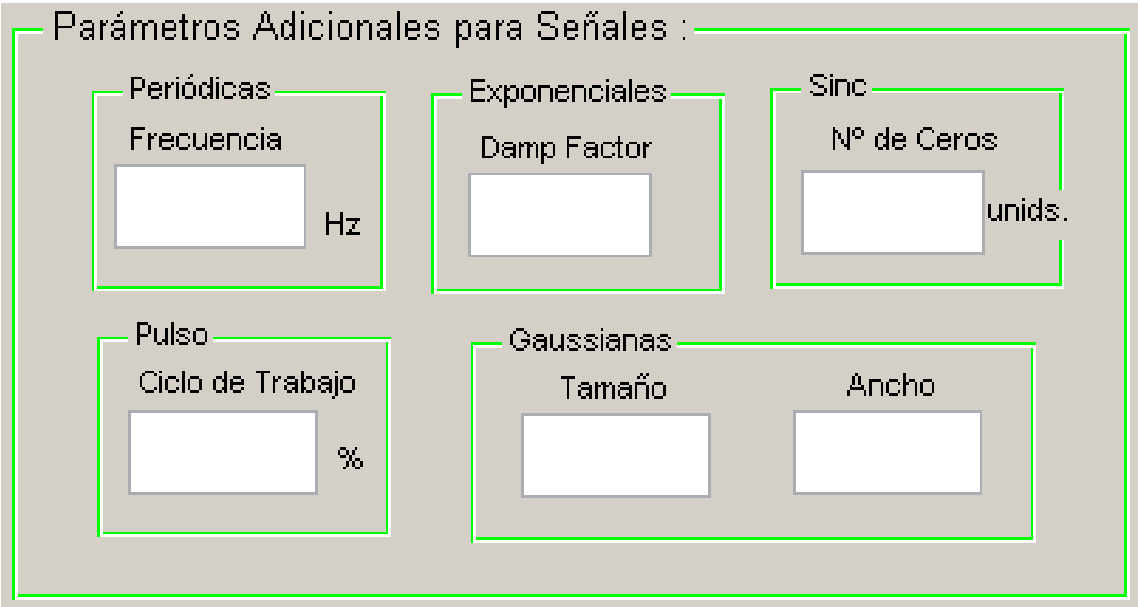


Figura 6.5. Panel de Parámetros adicionales a Introducir

El panel de creación de la señal también dispone de un botón “Generar Señal” el cual se encarga de graficar la señal diseñada.

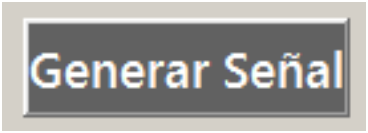


Figura 6.6. Botón Generar Señal.

Por último, para ofrecer ayuda al usuario que lo necesite, el Panel de Creación de la Señal dispone de un botón “Ayuda” el cual ofrece los pasos a seguir para el diseño de una forma de onda.

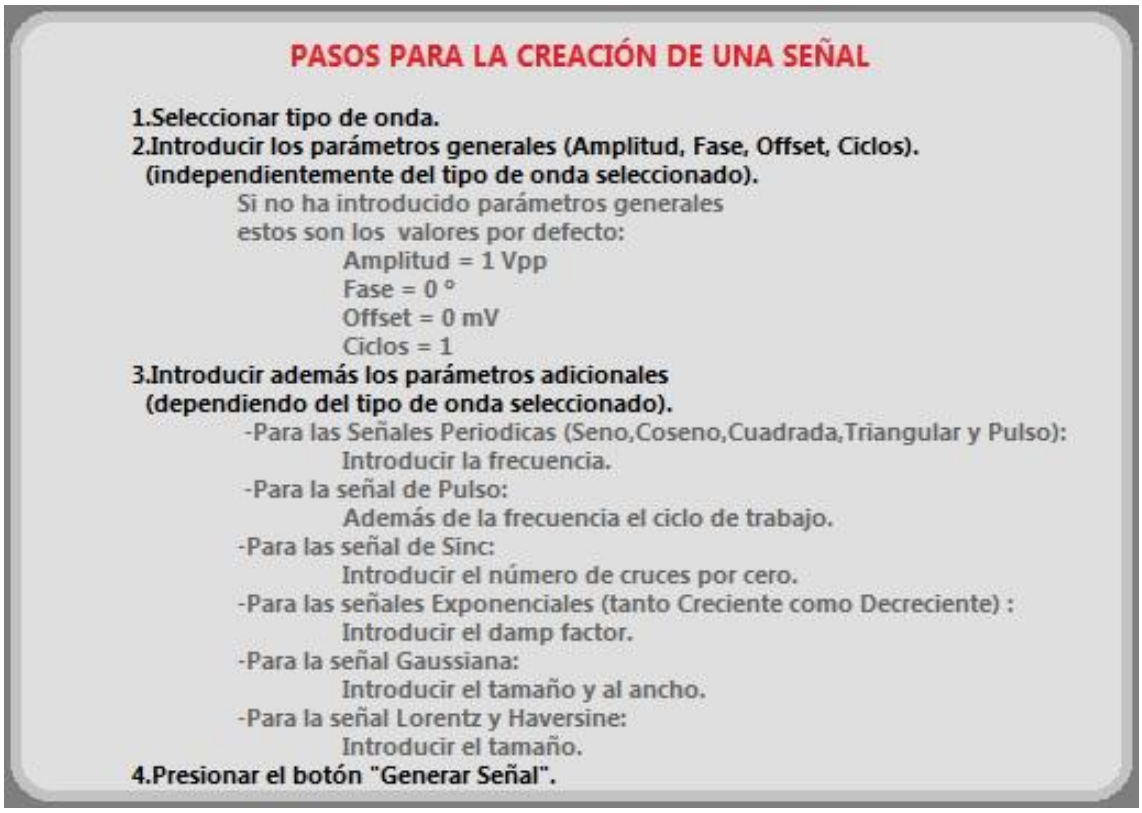


Figura 6.7. Pasos para la creación de una Señal.

Panel de la Gráfica seleccionada

Este panel se encarga de representar la forma de onda diseñada por el usuario.

En el eje X se representaran los valores del tiempo en segundos, y en el eje Y se representarán los valores de la amplitud de las señales en Voltios.

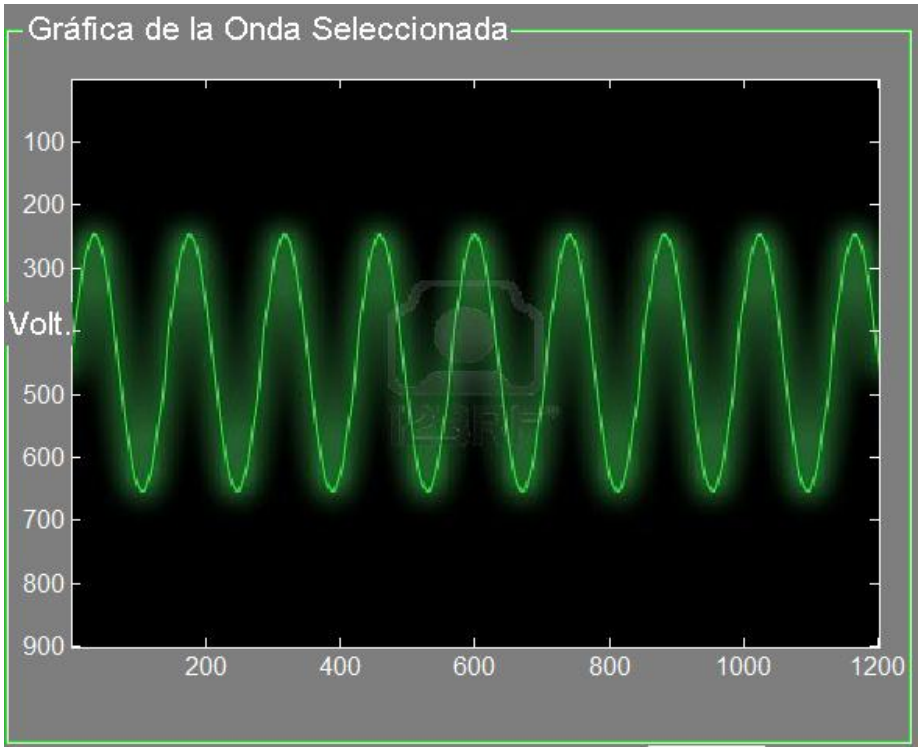


Figura 6.8. Gráfica de la Onda Seleccionada

Panel Enviar señal al Generador y Osciloscopio

Este panel se compone por dos opciones que sirven para elegir el canal de transmisión, el botón de "Enviar Señal "y otro botón de Ayuda.

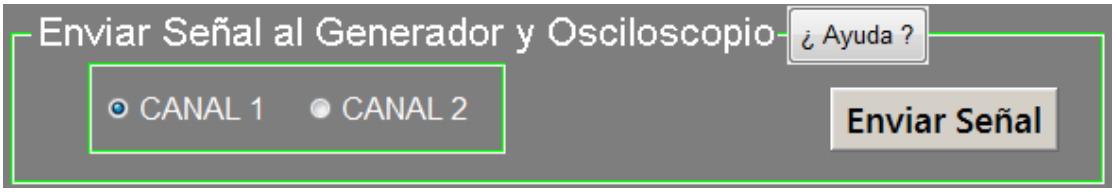


Figura 6.9. Panel para Enviar la Señal al generador

A continuación se muestra la pantalla generada al pulsar el botón de Ayuda, que muestra los pasos a seguir para la transmisión.

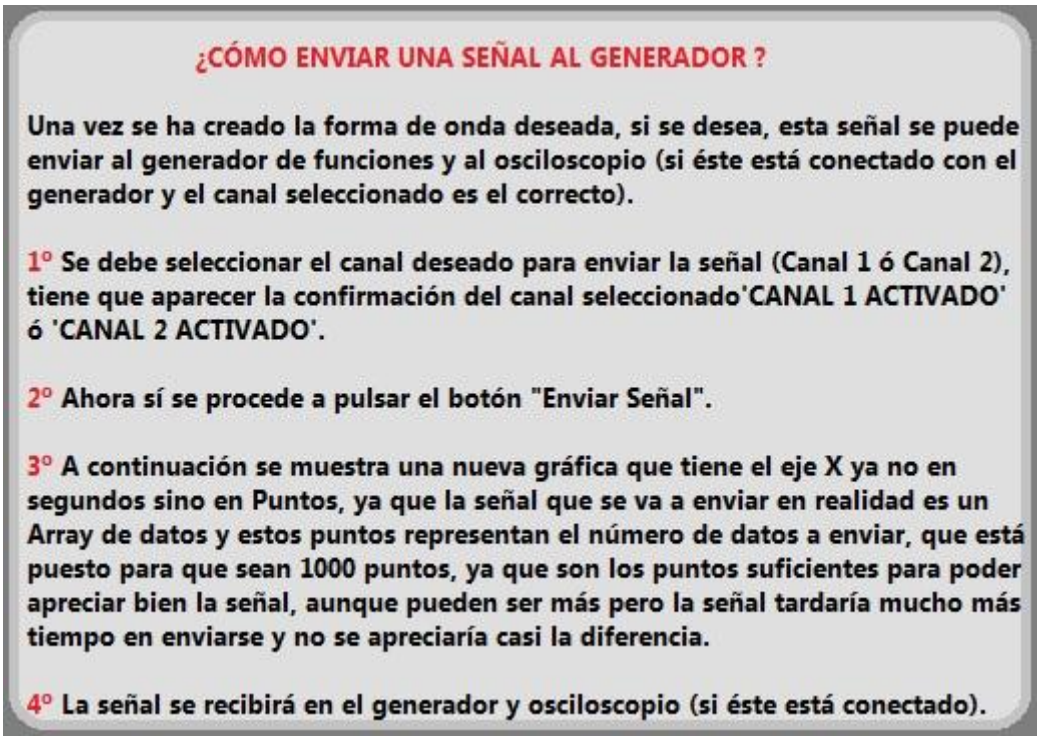


Figura 6.10. Ayuda para enviar una señal al generador.

6.3. Pasos a seguir para el funcionamiento del programa

Pasos previos:

1º Ver si el TekVisa Resource Manager ha detectado el generador y el osciloscopio, en el caso de que se desee conectar el osciloscopio con el generador de funciones.

2ºEjecutar el programa desde Matlab.

3ºUna vez que ya se ha abierto la interfaz gráfica, procedemos a diseñar la forma de onda.

Pasos para diseñar, graficar y enviar si se desea al generador de funciones una forma de Onda.

1º Seleccionamos el tipo de onda que se desea dibujar

2º Se introducen los parámetros generales, que son para todos los tipos de ondas, es decir, la Amplitud en Vpp (Voltios pico a pico), la Fase en grados, el Offset en mV (mili Voltios), y los ciclos, que son los periodos que se quieran dibujar de la señal.

- **NOTA: Si no se han introducido los parámetros generales éstos son los valores por defecto:**

  Amplitud = 1 Vpp

  Fase = 0 °

  Offset = 0 mV

  Ciclos = 1

3º Se introducen los parámetros adicionales, que varían dependiendo del tipo de onda a graficar.

- Si se desea dibujar una señal periódica como son el seno, coseno, triangular, cuadrada o pulso debemos indicar además la frecuencia.
- Para el pulso se debe introducir el Ciclo de trabajo (Duty Cicle), además de la frecuencia ya que también es periódica.
- Si se quiere dibujar una sinc se deben de introducir el número de cruces por cero.
- Si lo que se desea es dibujar una señal exponencial, tanto creciente como decreciente, tendremos que indicar el Damp Factor (Factor de Amortiguamiento).
- Para las gaussianas se deben de indicar su tamaño y su ancho.
- Y para las señales Lorentz y Haversine se deberán de introducir su tamaño.

- **NOTA: Si no se han introducido los parámetros adicionales necesarios para cada tipo de señal, se mostrará un mensaje de error indicando el parámetro a introducir.**

4º Se pulsa el botón Generar Señal.

5º Se observará la señal dibujada.

6º Si se desea enviar la señal al generador de funciones:

- Elegimos el canal de transmisión, y esperamos que se muestre un mensaje de confirmación del canal seleccionado 'CANAL 1 ACTIVADO' o 'CANAL 2 ACTIVADO'.
- Después pulsamos el botón “Enviar Señal”, también se enviará al osciloscopio si este se ha conectado con el generador y se ha seleccionado el canal correcto.
- A continuación se mostrará la misma gráfica de la señal pero con el eje X ya no en segundos sino en puntos, ya que la señal que se va a enviar en realidad es un array de datos y estos puntos representan el número de datos a enviar, que está puesto para que sean 1000 puntos, ya que son los puntos suficientes para poder apreciar bien la señal, aunque pueden ser

más, pero la señal tardaría mucho más tiempo en enviarse y no se apreciaría casi la diferencia.

7º Se recibirá la señal en el generador de funciones.

Ejemplo de funcionamiento:

1º Una vez comprobados los pasos previos, se elige la señal Seno.

2º Se introducen los parámetros generales: Amplitud=1Vpp, Fase=0 °, Offset=0 mV, Ciclos=2.

3º Ahora se introducen los parámetros adicionales: Frecuencia=1000 Hz.

4º Se pulsa el botón Generar Señal.

Figura 6.11. Ejemplo de Funcionamiento. Panel de creación de la señal.

Si no se hubieran introducido los parámetros adicionales, en este caso la frecuencia, se mostrará el siguiente mensaje de error:

Figura 6.12. Ejemplo Mensaje de error.

También se podía haber pulsado el botón de ayuda si se tenía alguna duda durante el proceso de creación de la señal.

5º Se observa la señal graficada.

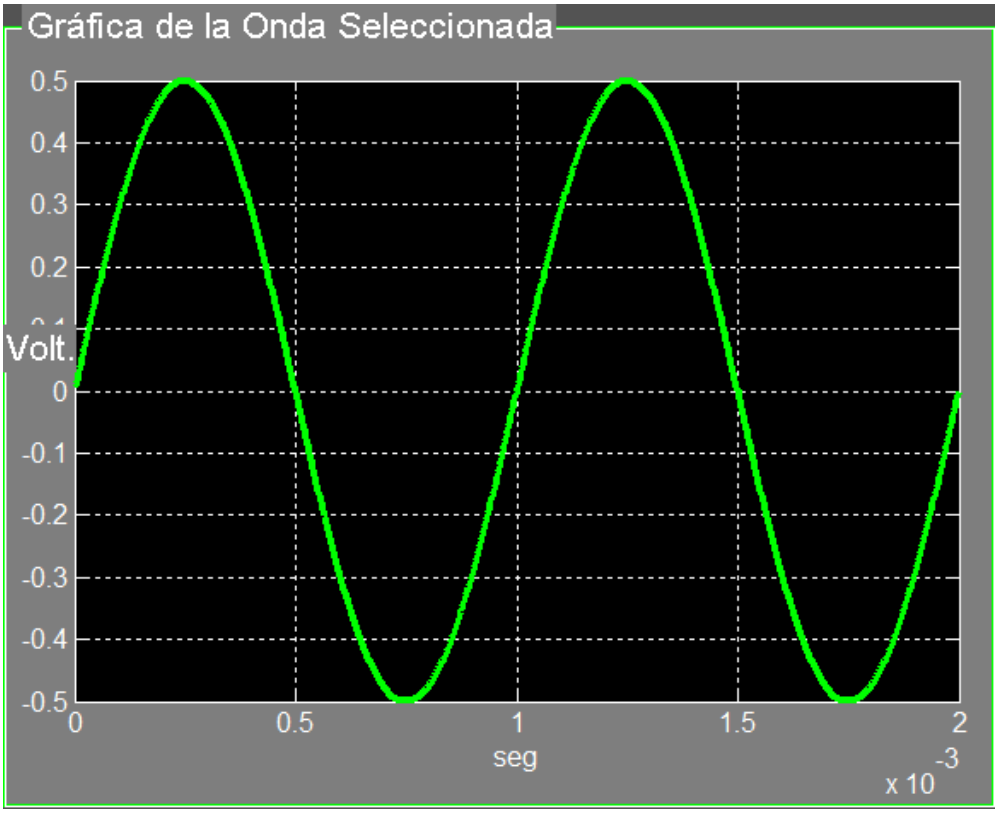


Figura 6.13. Ejemplo de Funcionamiento. Señal Seno representada.

6º Se pulsa el botón Enviar Señal

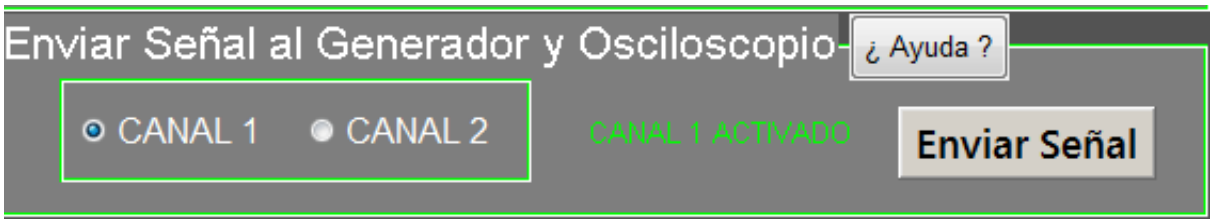


Figura 6.14. Ejemplo de Funcionamiento. Enviar Señal.

En caso de duda se puede pulsar el botón de "Ayuda" para averiguar cómo se envía una señal al generador.

7º Se recibirá la señal en el generador de funciones.

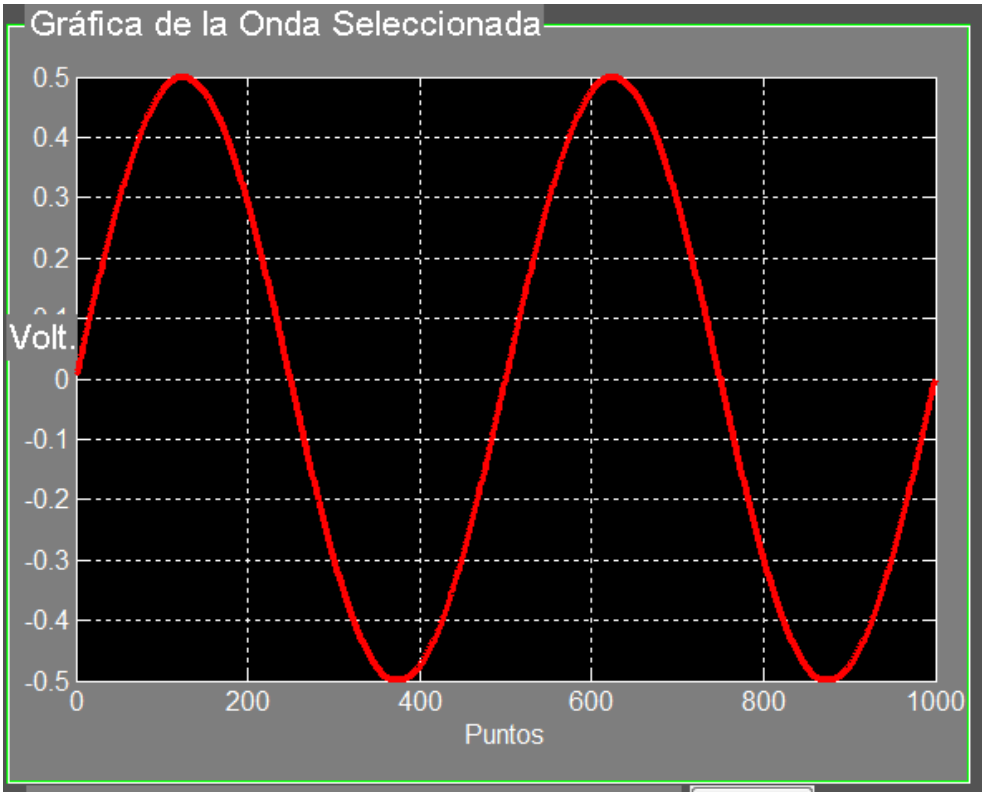


Figura 6.15. Ejemplo de Funcionamiento. Enviar Señal Seno.

6.4. Tipos de Ondas a Representar

A continuación se van a dibujar todos los tipos de señales periódicas como son el seno, coseno, cuadrada, triangular y pulso con los valores de parámetros de entrada indicados:

Parámetros Generales: Amplitud=1Vpp, Fase=0°, Offset=100mV y 2 ciclos de cada señal.

Parámetros adicionales para las señales periódicas: Frecuencia=1000Hz

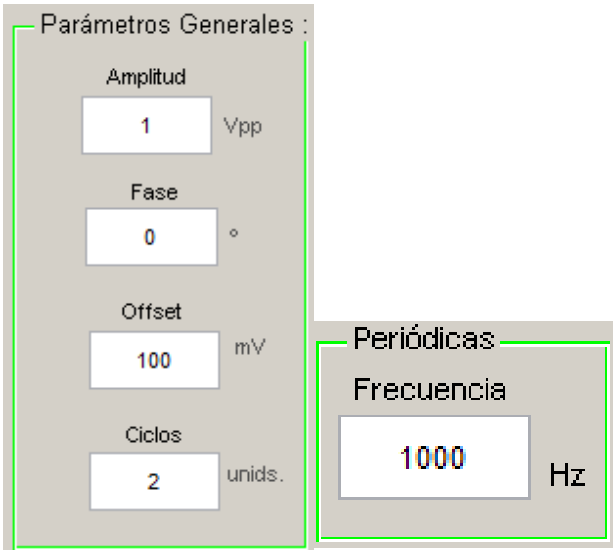


Figura 6.16. Parametros Introducidos para las señales periódicas

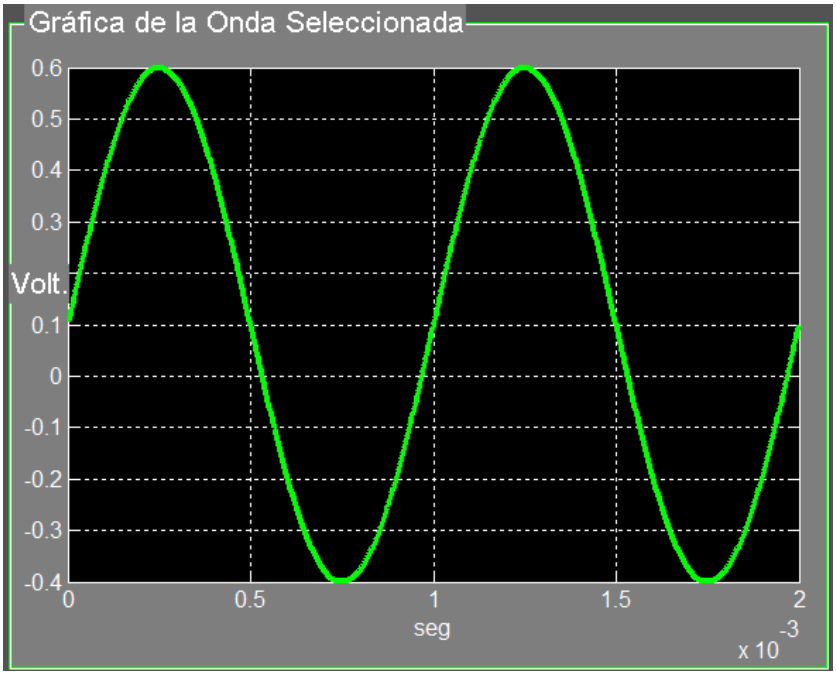


Figura 6.17. Gráfica de la señal Seno.

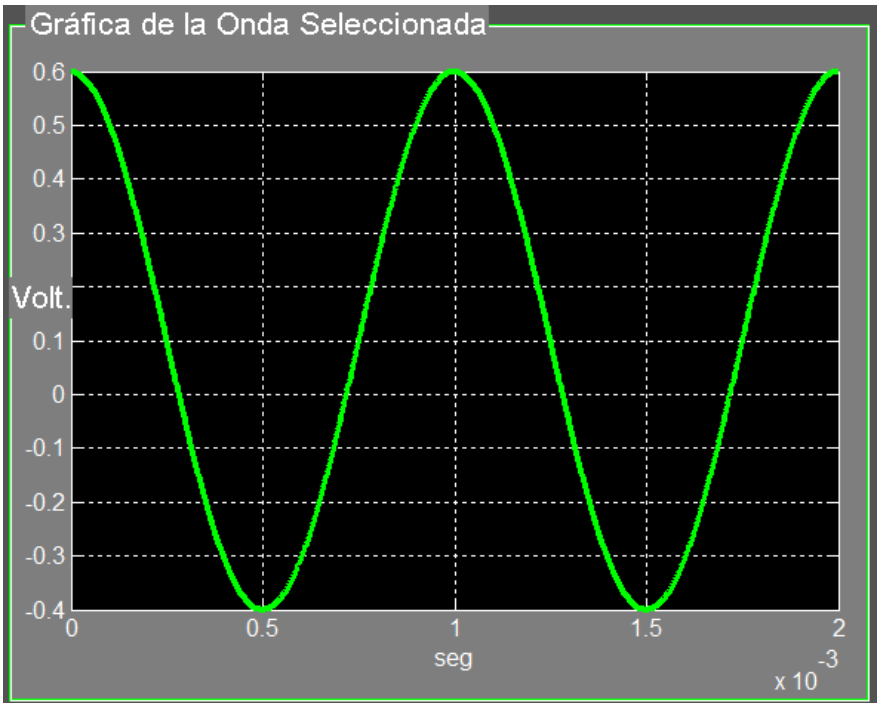


Figura 6.18. Gráfica de la señal Coseno.

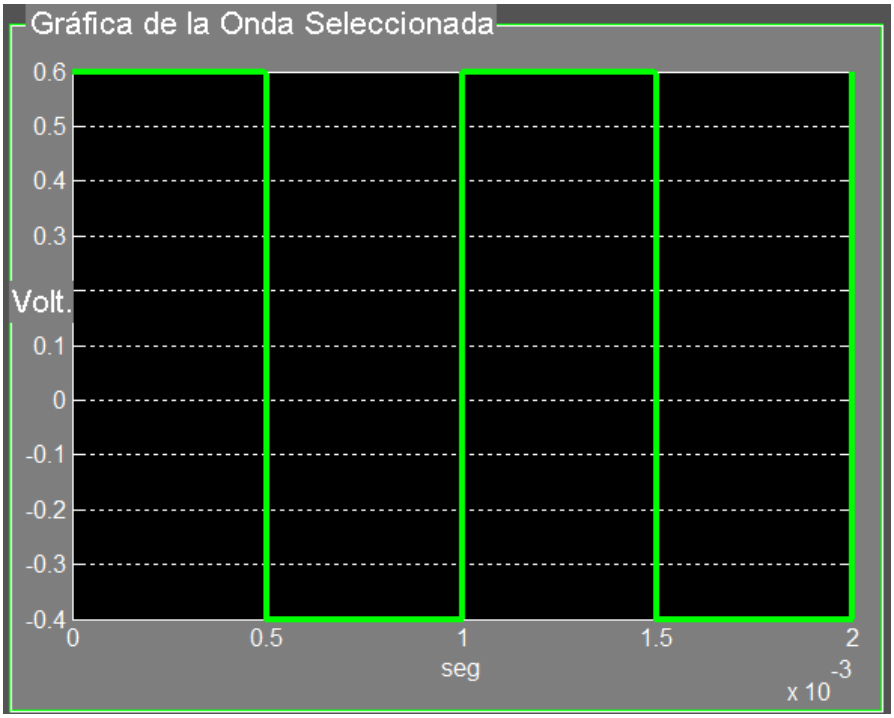


Figura 6.19. Gráfica de la onda cuadrada

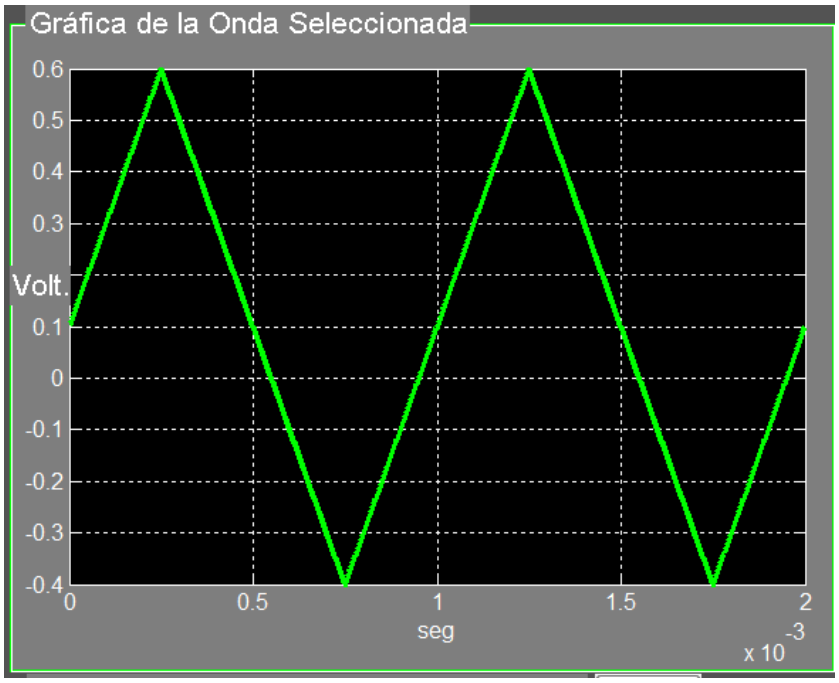
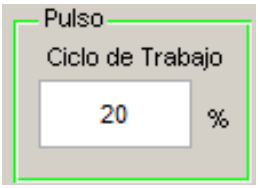


Figura 6.20. Gráfica de la onda triangular

Parámetros adicionales para el Pulso: Ciclo de Trabajo=20%



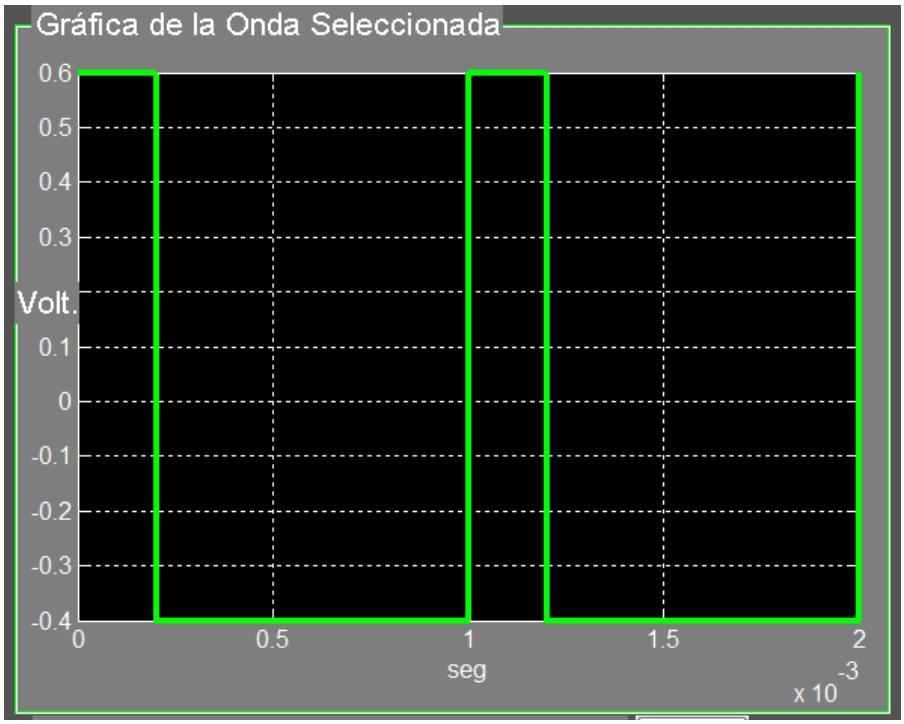


Figura 6.21. Gráfica de la señal de Pulso y el ciclo de trabajo.

Se dibujan las señales exponenciales con los siguientes parámetros generales y Damp Factor=0.5:

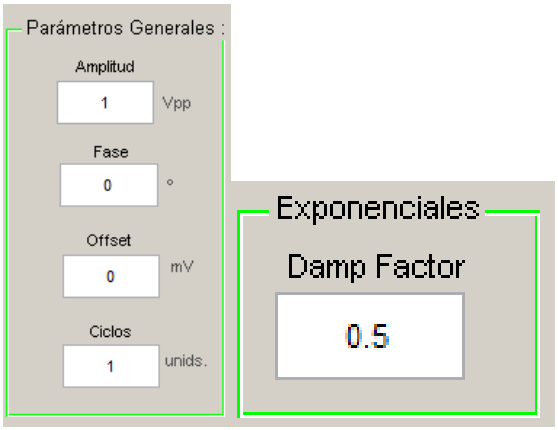


Figura 6.22. Parámetros introducidos para las señales exponenciales.

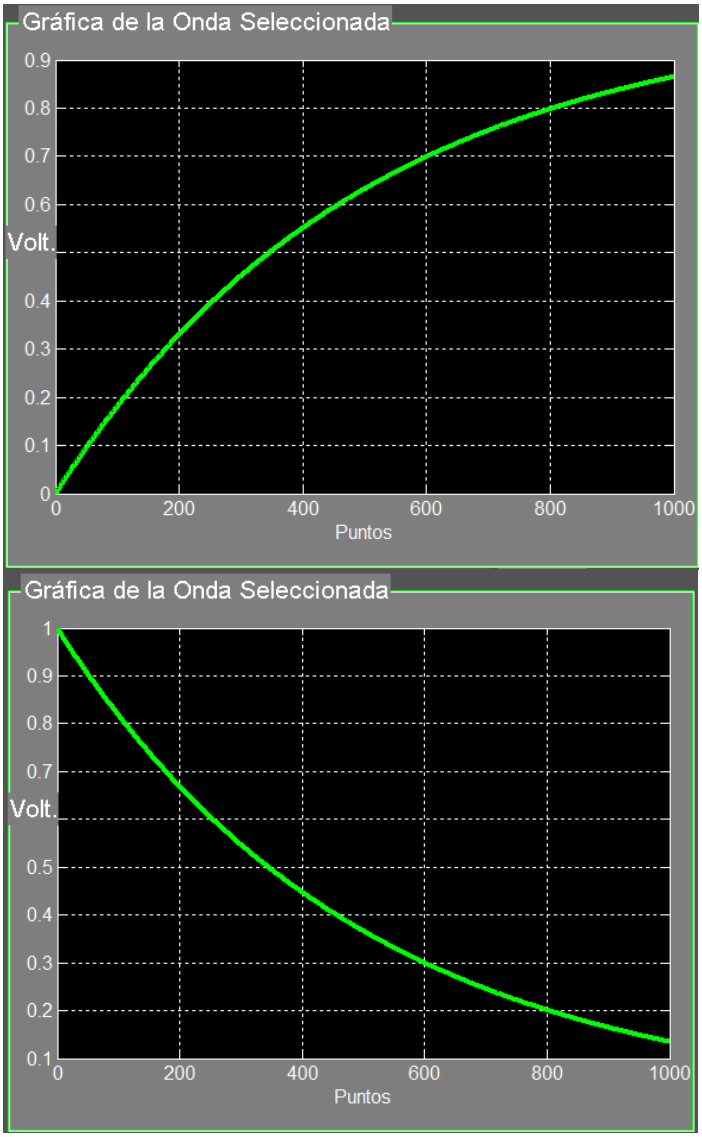


Figura 6.23. Arriba: Gráfica de la Exponencial Creciente y Abajo: Gráfica de la Exponencial Decreciente.

Ahora se crea la Sinc con los siguientes parámetros:

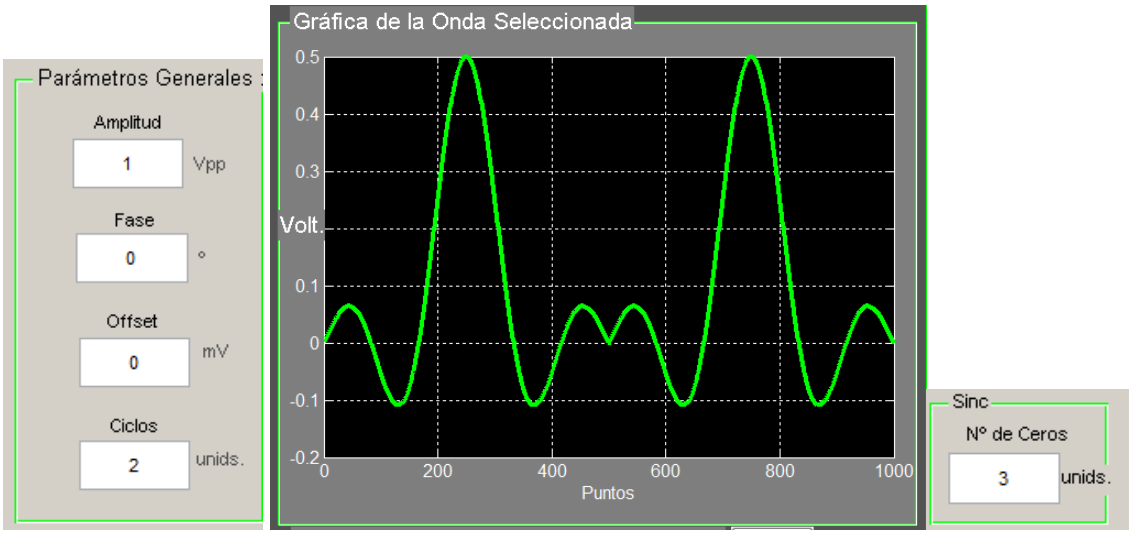


Figura 6.24. Grafica de la Sinc con sus parámetros introducidos

Creación de la señal DC con los siguientes parámetros:

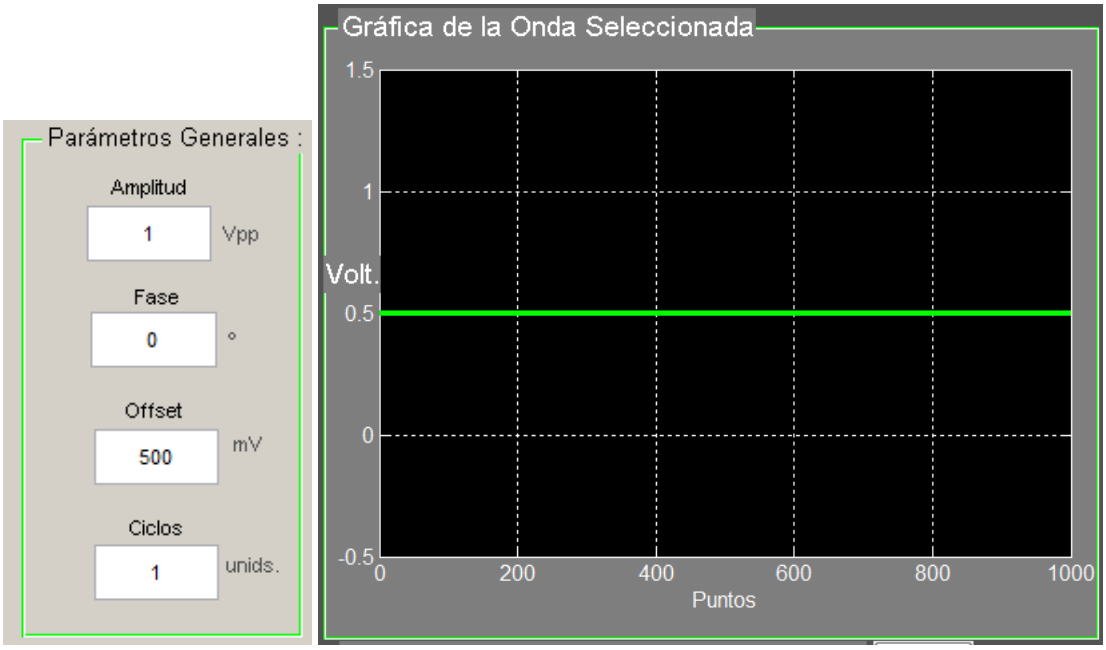


Figura 6.25. Grafica de la señal DC con sus parámetros introducidos.

Gráfica de la Gaussiana con los siguientes parámetros generales, que serán los mismos que se usarán para la señal lorentz y para la haversine.

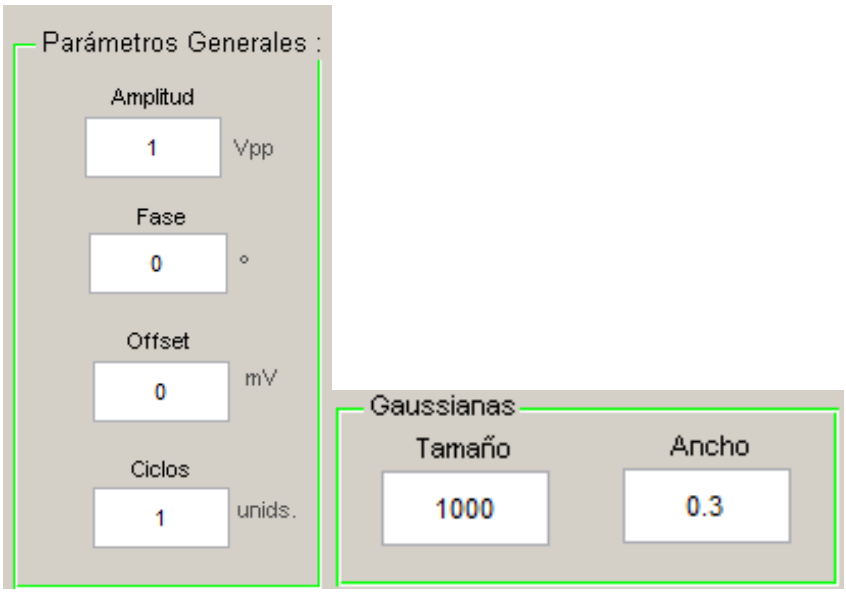


Figura 6.26. Parámetros para la señal Gaussiana.

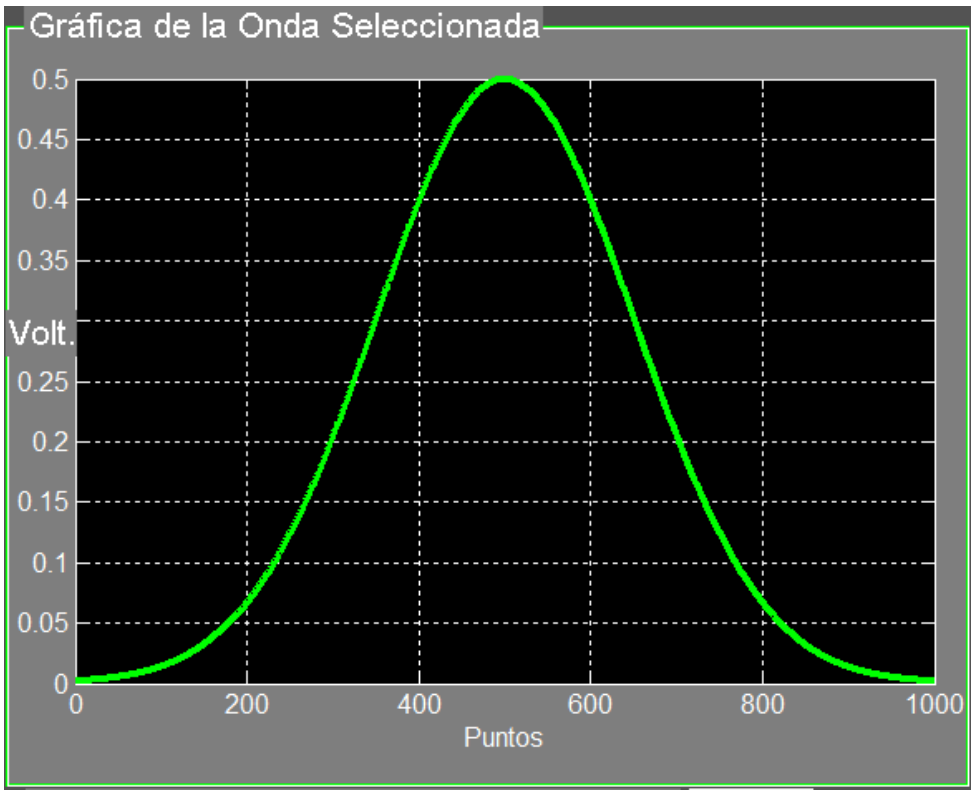


Figura 6.27. Grafica de la señal Gaussiana.

A continuación se presentan la señal de Lorentz y la señal de Haversine.

A éstas además de los parámetros generales solo se les podrá introducir el tamaño ya que ambas tienen el ancho predefinido.

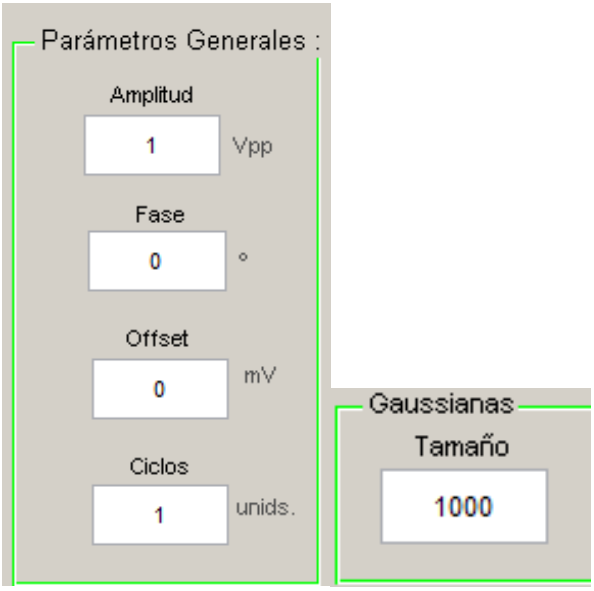


Figura 6.28. Parámetros para la señales Lorentz y Haversine.

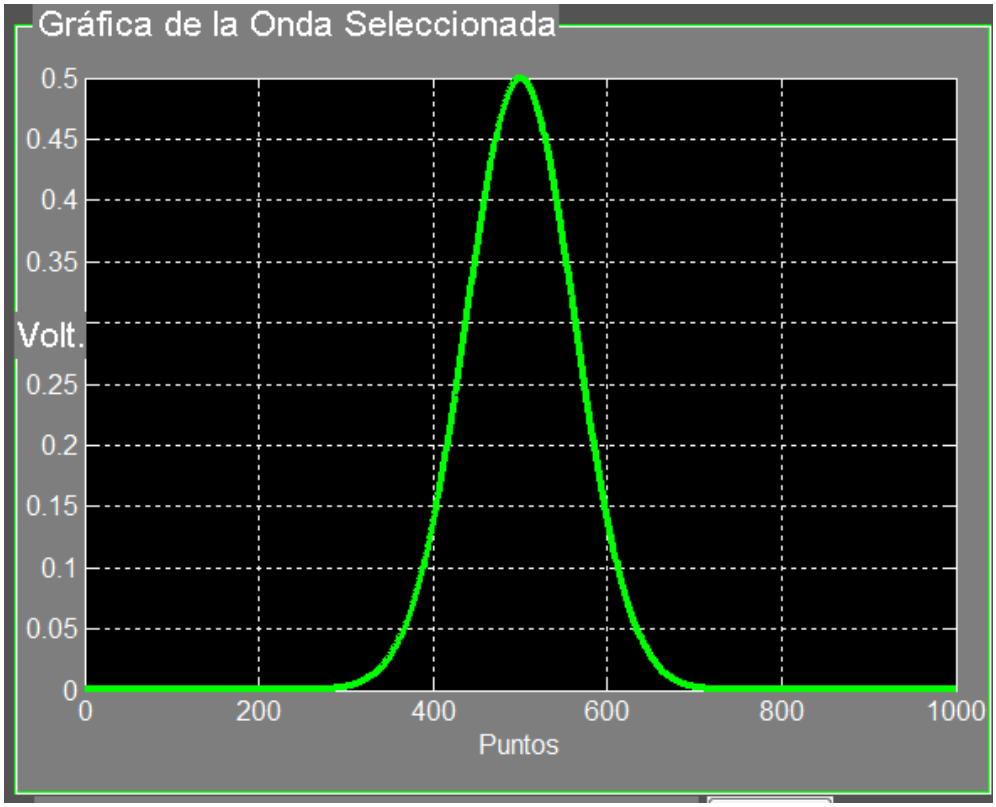


Figura 6.29. Grafica de la señal Lorentz.

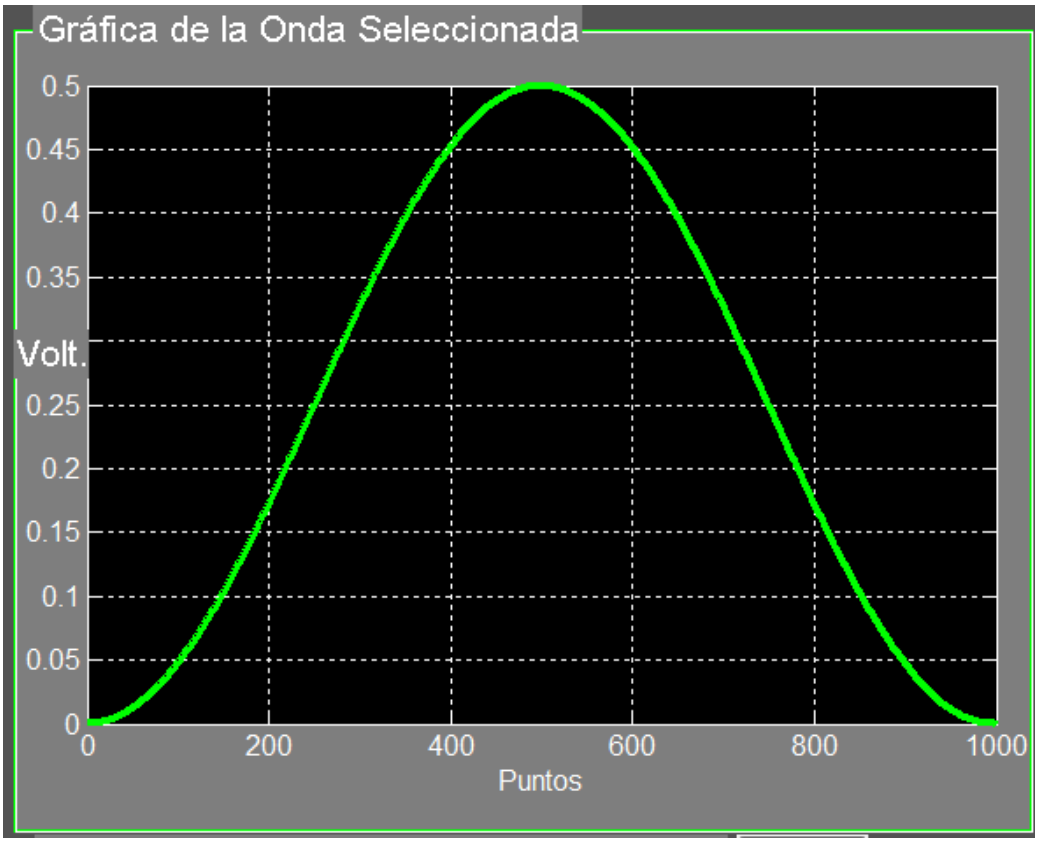


Figura 6.30. Grafica de la señal Haversine.

CAPÍTULO 7

Conclusiones

En este capítulo se pasará a resumir brevemente las consecuencias a las que se han llegado después de cumplir los objetivos que se planteaban al principio de este proyecto.

Después de haber estudiado la herramienta Tekvisa se ha logrado establecer una buena comunicación entre el generador y el pc a través de unos comandos creados por Tektronix para el software Matlab, en especial se han usado los de conexión vía LAN.

Después se han integrado estos comandos de comunicación con la interfaz gráfica desarrollada en Matlab, y con todo esto se ha logrado crear un programa completo en el cual se distinguen dos partes, primeramente el diseño y creación de la forma de onda y en segundo lugar la transmisión de esta señal al generador de funciones.

Con lo cual a través de esta interfaz gráfica se permite el control del generador directamente desde Matlab que es la herramienta que usan los alumnos a lo largo de sus carreras, en especial en las de telecomunicaciones.

Se ha conseguido crear un programa de fácil uso para el usuario.

Además se pretende introducir a los usuarios de este programa al estudio de las señales y ver cómo se comportan, también si se desea se podrán crear nuevas formas de onda e integrarlas con el resto del programa con lo que también se introduce al usuario en el campo de la programación y de la creación de interfaces gráficas.

CAPÍTULO 8

Presupuesto del Proyecto

En este último capítulo se presentan justificados los costes globales de la realización de este Proyecto Fin de Carrera.
Tales costes, imputables a gastos de personal y de material, se pueden deducir de las Tablas 8.1 y 8.2.
En la Tabla 8.1 se muestran las fases del proyecto y el tiempo aproximado para cada una de ellas.

Fase 1	Documentación	100 horas
Fase 2	Desarrollo del software	300 horas
Fase 3	Simulación y Análisis de los resultados	250 horas
Fase 4	Redacción de la memoria del proyecto	100 horas

Tabla 8.1 Fases del proyecto y el tiempo aproximado para cada una de ellas.

Así pues, se desprende que el tiempo total dedicado por el proyectando ha sido de 750 horas, de las cuales aproximadamente un 10% han sido compartidas con el tutor del proyecto directa o indirectamente. Por lo que el total asciende a 825 horas.
Teniendo en cuenta que la tabla de honorarios del Colegio Oficial de Ingenieros Técnicos de Telecomunicación establece unas tarifas de 60 €/hora, el coste de personal se sitúa en 49.500 €.

En la Tabla 8.2 se recogen los costes de material desglosados en equipo informático, software de simulación, documentación y gastos varios no atribuibles (llamadas telefónicas, desplazamientos...).

Ordenador de gama media	1.000€
Software de simulación	500€
Documentación	200€
Gastos Varios	500€

Tabla 8.2 Costes de material

Los costes de material ascienden a un total de 2.200€.

A partir de estos datos, el presupuesto total es el mostrado en la Tabla 8.3.

Concepto	Importe(€)
Costes personal	49.500
Costes material	2.200
Base imponible	51.700
I.V.A. (18%)	9.306
TOTAL	61.006

Tabla 8.3 Costes Total del Proyecto.

Con todo esto el coste total del proyecto es de 61.006 €.

El ingeniero proyectista,

Fdo. Paolo Horna Dueñas

CAPÍTULO 9

Referencias

[1] Manual de Usuario para los generadores de funciones/arbitrarios Tektronix® de la serie AFG3000.

[2] "TDS5000B Series Digital Phosphor Oscilloscopes Quick Start User Manual 071-1355-02" (Manual de Usuario de los Osciloscopios de Fósforo Digitales Tektronix® TDS5000B).

[3] "Programmer Manual TekVISA® 077-0140-00"

(Manual del programador TekVISA de Tektronix).

[4] "ArbExpress® AXW100 Waveform Creation and Editing Tool for Tektronix AWG/AFG Version 2.4, 077-0000-04"

(Arbexpress AXW100 Creación de Formas de onda y herramienta de edición para generadores Tektronix AWG/AFG)

[5] www.tek.com

[6] www.wikipedia.org

[7] "Manual de Interfaz Gráfica de Usuario en Matlab (parte 1)"

Autor: Diego Orlando Barragán Guerrero.

www.matpic.com